LENGTHENING THE TIMESCALE REACH OF MOLECULAR DYNAMICS

BY

DAVID ELDON TANNER

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Biophysics and Computational Biology
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2012

Urbana, Illinois

Doctoral Committee:

Professor Klaus J. Schulten, Chair
Associate Professor Claudio Grosman
Professor Laxmikant Kale
Professor Colin A. Wraight

Abstract

Molecular dynamics (MD) is a computational method employed for studying the dynamics of nanoscale biological systems on nanosecond to microsecond timescales. Using MD, researchers can utilize experimental data from crystallography and cryo-electron microscopy (cryo-EM) to explore the functional dynamics of biological systems. The timescale reach of the molecular dynamics tool is limited by how fast femtosecond timesteps can be sequentially integrated; today's fast computers allow simulation speeds of tens of nanoseconds of simulation time per day, which typically limits simulation lengths to hundreds of nanoseconds. This work explores three ways whereby the timescale reach of molecular dynamics can be lengthened beyond nanoseconds, to the millisecond timescales of cellular processes. First, a theoretical model of flagellin translocation allows nanosecond timescale MD simulations to explore the hour-long process of bacterial flagellum elongation. Second, a generalized Born model of implicit solvent accelerates simulation through reduced computational expense as well as increased conformational sampling due to reduced viscosity of the implicit solvent. Finally, advanced computing technologies, such as graphics processing units, accelerate simulation speeds of hybrid GB/SA implicit solvent models, thereby directly increasing simulation lengths.

To my wife, JaNae,

who strove to accomplish as much in her

family, religious and community service,

as I in my doctoral research.

Thank you for your love and patience.

Acknowledgements

I would first like to thank all the co-authors on the publications included in this work: Yan Chan, Wen Ma, Zhongzhou Chen, Jim Phillips and Klaus Schulten. Additionally, I thank colleagues who helped through insightful discussion and software implementation advice: Johan Strumpfer, Anton Arkhipov, Peter Freddolino, Chris Harrison, John Stone and Sanjay Kale. This work was supported by the National Science Foundation (NSF PHY0822613 to K. Schulten), the National Institutes of Health (NIH P41-RR05969 to K. Schulten and Molecular Biophysics Training Grant fellowship to D. Tanner), and the Computational Science and Engineering fellowship at UIUC to D. Tanner. Computer time was provided through the National Resource Allocation Committee grant (NCSA MCA93S028 to K. Schulten) from the National Science Foundation and the Extreme Science and Engineering Discovery Environment (XSEDE), which is supported by National Science Foundation grant number OCI-1053575.

Table of Contents

Chapter 1

Introduction

Molecular dynamics (MD) is a computational method [1] employed for studying the dynamics of nanoscale biological systems on nanosecond to microsecond timescales [2]. Using MD, researchers can utilize experimental data from crystallography and cryo-electron microscopy (cryo-EM) to explore the functional dynamics of biological systems [3].

Molecular dynamics has already enabled modeling of biomolecular systems and processes, e.g., viral infection [4, 5], interactions between diseases and therapeutics [6, 7], blood coagulation [8–15], and amyloid fibril formation [16–21]. Only by continually adopting latest methodologies and technologies can researchers extend the reach of MD to the millisecond-scale [22] cellular processes, vital to life.

The timescale reach of the molecular dynamics tool is limited by how fast femtosecond timesteps can be sequentially calculated; today's fast computers allow simulation speeds of tens of nanoseconds of simulation time per day which typically limits simulation lengths to hundreds of nanoseconds. This work explores three ways whereby the timescale reach of molecular dynamics can be lengthened, beyond nanoseconds, to cellular timescales. First, a theoretical model of flagellin translocation allows nanosecond timescale MD simulations to explore the hour-long process of flagellum elongation. Second, a generalized Born model of implicit solvent accelerates simulation through reduced computational expense as well as increased conformational sampling due to reduced viscosity of the implicit solvent. Finally, advancing computing technologies, such as graphics processing units, accelerates simulation speeds which increases simulation lengths.

1.1 Theoretical Model of Flagellin Translocation

The bacterial flagellum is a self-assembling filament, which bacteria use for swimming [23–26]. It is built from tens of thousands of flagellin monomers in a self-assembly process that involves translocation of the monomers through the flagellar interior, a channel, to the growing tip [27, 28]. Flagellum monomers are pumped into the filament at the base, move unfolded along the channel and then bind to the tip of the filament, thereby extending the growing flagellum [29–33]. The flagellin translocation process, due to the

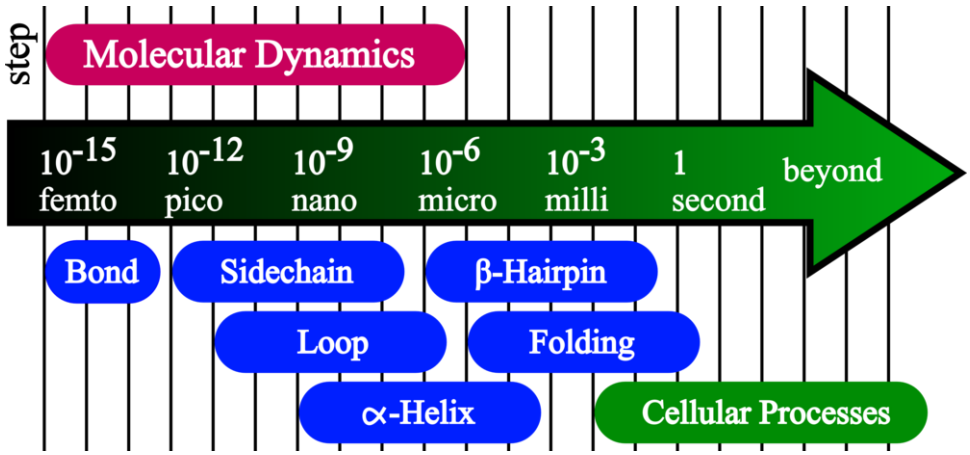

Figure 1.1: Because modern computers can only calculate millions of femtosecond timesteps per day, molecular dynamics simulations typically only reach timescales of hundreds of nanoseconds. While nanosecond-scale simulations are able to probe biological phenomena such as simple protein folding, they cannot probe the longer timescales of cellular processes.

flagellum maximum length of 20 μm, is an extreme example of protein transport through channels.

A straightforward approach to the study of flagellum elongation might be an all atom simulation of translocating flagellin and the elongating filament; however, the large size of the flagellum and the long time scale of growth render this approach intractable. Therefore, another strategy is adopted, which starts from a mathematical model of the translocation-elongation process; the model describes all properties influencing the process rate, including flagellin density, compressibility and friction, and reproduces the observation reported in [27] as well as overall filament length. The local physical properties underlying the model are then verified through molecular dynamics simulations, the model thus furnishing a bridge between small molecule-short time scale properties of the translocating flagellin accessible to molecular dynamics and the large size-long time behavior of the growing flagellum accessible to experiment [27].

1.2 Implicit Solvent

Because biological processes take place in the aqueous environment of the cell, a critical component of any biological MD simulation is the solvent model employed [34, 35]. An accurate solvent model must reproduce water's effect on solutes such as the free energy of solvation, dielectric screening of solute electrostatic interactions, hydrogen bonding and van der Waals interactions with solute.

An alternative representation of the aqueous environment of the cell is furnished by implicit solvent descriptions which eliminate the need for explicit solvent molecules. The generalized Born implicit solvent (GBIS) model, used by MD programs CHARMM [36, 37], Gromacs [38, 39], Amber [40] and NAMD [41–43],

furnishes a fast approximation for calculating the electrostatic interaction between atoms in a dielectric environment described by the Poisson-Boltzmann equation. The absence of explicit water molecules also eliminates the viscosity imposed on simulated solutes, allowing faster equilibration of solute conformations and better conformational sampling, thus lengthening the effective simulation timescale.

To illustrate the utility of NAMD's parallel GBIS implementation and how the use of implicit solvent can accelerate simulations and lengthen the simulation timescale, we simulate the *Escherichia coli* ribosome. The ribosome is the cellular machine that translates genetic information on mRNA into protein chains. During protein synthesis, the ribosome complex fluctuates between two conformational states, namely the so-called classical and ratcheted state [44]. During transition from classical to ratcheted state, the ribosome undergoes multiple, large conformational changes, including an inter-subunit rotation between its 50S and 30S subunits [44] and the closing of its L1 stalk in the 50S subunit [45] (see 3.4).The large conformational changes during the transition from classical to ratcheted state are essential for translation [46] as suggested by previous cryo-EM data [47]. To demonstrate the benefits of NAMD GBIS, we simulate the large conformational changes during ratcheting of the ∼250,000-atom ribosome using molecular dynamics flexible fitting.

1.3 Advanced Computing Technology: the Graphics Processing Unit

Though originally developed for high-quality rendering, graphics processing units (GPUs) are well suited for accelerating parallel MD simulations. A single GPU may contain 16 multiprocessors (MP), each with 32 cores, making a 512-core processor which can be much more powerful than typical 8 or 16-core CPU processors. GPUs have already accelerated biological computing applications such as: molecular modeling [48], electrostatics [49], molecular orbital display [50], simulations of protein diffusion in whole cells [51].

An ideal application for computing with GPUs and CPUs is the generalized Born (GB) implicit solvent calculation with solvent-accessible surface area (SA) calculation, commonly abbreviated GB/SA. A GB/SA implicit solvent simulation employs the GB calculation to account for the polar, i.e., hydrophilic, effects of water and the SA calculation to model the nonpolar, i.e., hydrophobic, effect of water [52]. While GB force calculation is amenable to the GPU, certain characteristics of the LCPO's SA calculation make CPU calculation more suitable.

A new hybrid algorithm for GB/SA simulations on heterogeneous GPU/CPU computers allows for faster and more efficient MD simulations. Using the modern computational technologies such as the GPU is

necessary for achieving fast simulations. As it often not feasible nor ideal to perform every necessary calculation on the GPU, it becomes increasingly important to develop methods for overlapping GPU and CPU calculations. The present work demonstrates an efficient method for performing GB/SA simulations on hybrid CPU/GPU computers, with the GB [53] calculation being performed on the GPU while the SA is calculated by the linear combination of pairwise overlaps [54] method. By incorporating GPUs into MD calculations, simulation speeds are advanced and longer timescale simulations can be achieved, as demonstrated through ∼250 benchmark simulations.

Chapter 2

Theoretical and Computational Investigation of Flagellin Translocation and Bacterial Flagellum Growth

2.1 Introduction

In many living systems, proteins need to be transported from where they are synthesized to where they are needed. Protein transport often requires unfolded proteins to pass through narrow pores or channels. Examples of protein transport systems are SecY [55] or the mitochondrial TOM (translocase of outer membrane) and TIM (translocase of inner membrane) [56]. Translocation can be driven by passive diffusion [57] or by active, i.e., energy consuming, transport [58]. Much theoretical research has focused on protein translocation through pores [57, 59–62] and channels [63, 64]. The bacterial flagellum, which allows bacteria to propel themselves [23–26], is a unique class of protein transport systems in that the conducting channel is considerably longer than the translocated protein [65].

The flagellar filament is typically $L = 10 - 20$ μm long and is built from tens of thousands of flagellin monomers stacked in a helical pattern leaving an interior space, the channel, as shown in Fig. 2.1 [27, 28]. Each repeat of the helix involves 11 monomers and a rise of 52 Å. The flagellin protein, PDB code 1UCU [28], consists of 494 amino acids. Fig. 2.1 shows the structure of the four flagellin domains, D0 (residues 1-55, 451-494), D1 (residues 56-176, 402-450), D2 (residues 177-189, 284-401), and D3 (residues 190-283) [28, 66, 67]. CD0 (residues 457-494) is an α-helix at the C-terminus, which comprises the inner surface of the filament channel that has a radius of $R = 10$ Å [28].

The flagellum is a self-assembling system. In the basal body, which anchors the rotating flagellum to the cell membrane [68], a type III secretion system pumps newly synthesized, unfolded, flagellin monomers into the flagellum channel [29–33]. Each added monomer displaces it's distal neighbor toward the tip of the filament, the secretion pump providing the driving force to translocate all flagellins in the channel toward

Material in this chapter is reproduced in part with permission from David E. Tanner, Wen Ma, Zhongzhou Chen and Klaus Schulten, "Theoretical and computational investigation of flagellin translocation and bacterial flagellum growth," *Biophysical Journal*, 100:607-614 (2011).

the growing tip. The flagellin monomers translocate in an unfolded conformation through the filament to the distal end of the flagellum, where they fold into an active conformation and bind to the filament with the help of a cap protein, thereby extending the flagellum [69]. Though both flagellum and translocating protein are comprised of flagellin, the term flagellin will be used here to denote only the translocating, unfolded, protein.

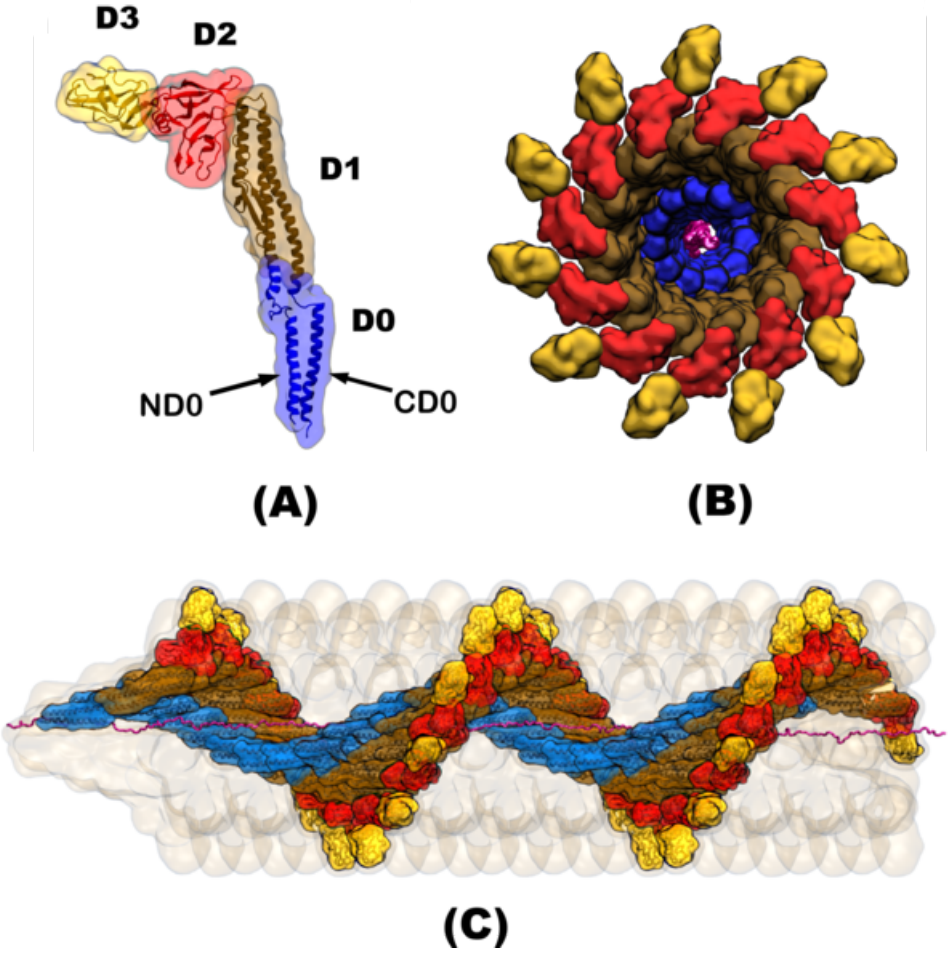

Figure 2.1: Flagellum structure. The flagellum is built from flagellin monomers stacked in a helical pattern. (A) The folded flagellin consists of four domains: D0 (blue), D1 (brown), D2 (red) and D3 (gold). (B) View along the filament axis shows how the domains are ordered by their distance from the channel's center. D0 is the inward most domain. CD0 is the α-helix at the C terminus of the D0 domain; it comprises the channel's inner surface. (C) Side view, with most of the filament monomers removed, shows how an unfolded flagellin monomer (magenta) spreads through the filament channel at near maximal extension.

Regarding flagellum growth, Iino demonstrated that the rate of filament elongation *in vivo* decays exponentially with length, while the growth rate *in vitro* (allowing flagellin in bulk water to bind directly to the flagellum tip without traversing the flagellum) is constant [27]. The *in vivo* decrease in rate was shown to result from a decrease in translocation efficiency as the filament grows. Levy sought to describe the flagellum

growth rate in terms of a concentration gradient of a flagellar binding factor [70, 71]. This model, developed before the channel structure had been solved [25], was based on the assumption that flagellin could freely diffuse from the base to the tip and did not correctly reproduce the exponential decay in the growth rate [27]. We explain the observed filament growth rate and its length-dependence in terms of the molecular properties of unfolded, i.e., translocating, flagellin and of folded flagellin, helically assembled into a flagellum.

A straightforward approach to the intended study might be an all atom simulation of translocating flagellin and the elongating filament; however, the large size of the flagellum and the long time scale of growth render this approach intractable. The authors, rather, adopt a strategy, which starts from a mathematical model of the translocation-elongation process; the model describes all properties influencing the process rate, including flagellin density, compressibility and friction, and reproduces the observation reported in [27] as well as overall filament length. The local physical properties underlying the model are then verified through molecular dynamics simulations, the model thus furnishing a bridge between small molecule-short time scale properties of the translocating flagellin accessible to molecular dynamics and the large size-long time behavior of the growing flagellum accessible to experiment [27].

Below, we present the theoretical description of flagellin translocation, which includes several assumptions that are either assumed self-evident or verified through simulation. Next, we outline the molecular dynamics simulations employed to verify the assumptions made and to furnish key system parameters needed to calculate the flagellum growth rate.

2.2 Methods

Our mathematical description of flagellin translocation is based on a minimum set of rather self-evident assumptions and on certain mechanical properties. The model is shown to agree well with observed growth rates, in particular, their length dependence. Molecular dynamics simulations carried out to test the relationship between model and molecular characteristics are then described.

2.2.1 Theory of Flagellin Translocation and Flagellum Growth

Our description of translocating unfolded flagellin is based on five position-dependent properties. These properties depend on their distance from the channel tip, x, and on the length of the flagellum, L. We emphasize that x denotes the distance from the tip, not from the base. The five characteristics are (1) flagellin density, $\rho(x, L)$ (in units flagellin/volume); (2) flagellin pressure, $p(x, L)$ (in units force/area); (3) friction density, $g(x, L)$ (in units force/area); the friction is due to atomic interactions at the contact surface

between flagellin and channel; (4) flagellin flux, $\phi(x, L)$ (in units flagellin/(time area); (5) translocation velocity, $v(x, L)$ (in units length time).

The theoretical description begins with a derivation of flagellin pressure, $p(x, L)$. It then uses this expression to derive expressions for the remaining four flagellin properties. Finally, the flagellum growth rate is calculated from these properties. The derivation is based on several conditions, which the translocating flagellin system must meet in order for the model to apply. These conditions are stated and are either argued to be self-evident or are verified through molecular dynamics simulations.

We start by postulating

Condition 1 *Flagellin pressure, p, is isotropic.*

The radial pressure which unfolded flagellin exerts against the channel surface should be equal to the axial pressure which it exerts on its proximal and distal neighbors,

$$p_{\text{axial}} = p_{\text{radial}} \,. \tag{2.1}$$

Indeed we expect that a denatured protein, a self-avoiding but otherwise rather flexible polymer, possesses this property typical for a fluid and does relay compression, exerted on it axially, also radially.

The next two postulates,

Condition 2 *Friction density, g, is proportional to flagellin pressure, p,*

and

Condition 3 *Translocating flagellin is in mechanical equilibrium,*

are closely related and rely on the fact that the maximum translocation velocity in the flagellar channel is on the order of 1 nm/ms while the maximum growth rate, V_0, is ~ 0.1 Å/ms [27]; this slow velocity is central to the behavior of friction between a flagellin and the channel. For velocities below a critical velocity, v_{c} (typically $\sim$1 nm/ms), friction is not characterized by a velocity-dependent dynamic friction, but rather by velocity-independent static friction [72, 73], known as "creeping" static friction [74, 75]. Because of the small driving forces and slow translocation rates present in the flagellum, flagellar friction is due to creeping static friction [76]. The flagellin-channel friction manifests itself through resistance to lateral (along channel axis) force which, over a segment $[x + dx, x]$, is due to the pressure difference at $x + dx$ and x and measures $f_{\text{lateral}} = \pi R^2 p(x + dx, L) - \pi R^2 p(x, L)$. The resisting force, creeping static friction, is due to adhesion of flagellin to the channel surface, $2\pi R\, dx$, and is stated the form $f_{\text{resistance}} = 2\pi R g(x, L) dx$ where $g(x, L)$ is the static friction per unit contact area. Both channel surface and translocating flagellin are

fluctuating conformationally, such that adhesion interactions are constantly formed and broken; as a result f_{lateral} imposes a bias that the flagellin re-forms its contacts with the channel further toward the distal tip; $f_{\text{resistance}}$ arises from the number of adhesion spots averaged along the channel and through time. This average is proportional to the number of flagellin surface atoms pressing against the channel wall and, hence, is proportional to the isotropic pressure of flagellin. Since translocation is so slow, Condition 2 cannot be tested through simulation. However, we demonstrate below that in the pressure and density ranges relevant for flagellin transport, pressure and surface contact density (number of contacts exposed to channel surface per unit surface area), namely $N_{\text{contact}}/A_{\text{contact}}$, are linearly related,

$$p_{\text{axial}} \;\sim\; N_{\text{contact}}/A_{\text{contact}} \; . \tag{2.2}$$

We express Condition 2 through

$$g = \alpha \, p \tag{2.3}$$

where α is a dimensionless proportionality coefficient, and Condition 3 through

$$\pi R^2 p(x + dx, L) \;-\; \pi R^2 p(x, L) \;-\; 2\pi R \, g(x, L) dx \;=\; 0 \; . \tag{2.4}$$

For $dx \to 0$, this reads

$$\partial_x \, p(x, L) \;=\; (2/R) g(x, L) \; . \tag{2.5}$$

Equation 2.5 and Eq. 2.3 combine to the differential equation

$$\partial_x \, p(x, L) \;=\; (2\alpha/R) p(x, L) \; , \tag{2.6}$$

the solution of which is

$$p(x, L) \;=\; p_0(L) \exp[(2\alpha/R)x] \; . \tag{2.7}$$

In Eq. 2.7, $p_0(L)$ is the flagellin pressure at the tip of the flagellum that pushes flagellin against the cap protein and is controlled through the non-zero force, f_{tip} [69], with which the cap protein resists flagellin to pass through the tip, such that the condition

$$p_0(L) \;=\; f_{\text{tip}}/\pi R^2 \tag{2.8}$$

holds. Accordingly we assume

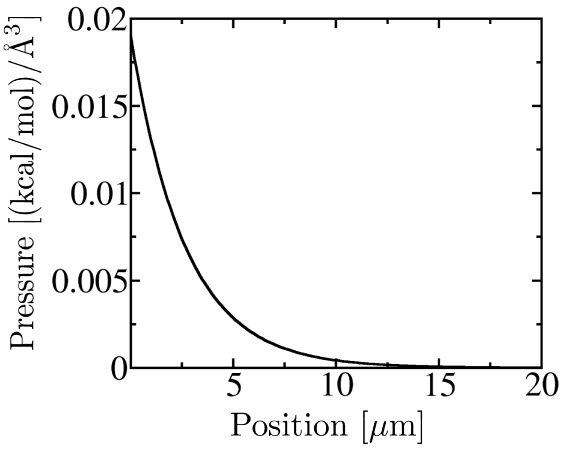

Figure 2.2: Flagellin Pressure. Using the system parameters listed in Table 2.2, including $P = 100$ (kcal/mol)/s, with Eq. 2.9 of the main text, flagellin pressure is shown as a function of position along the flagellum for the case $L = 20\mu$m. Pressure is highest at the flagellum base (position = 0) and lowest at the the flagellum tip (position = 20 μm). This plot reveals the flagellin pressure range relevant to the flagellum, namely $[1 \times 10^{-5}, 0.02]$ (kcal/mol)/Å^3.

Condition 4 *Flagellin pressure at the flagellar tip is* $f_{\rm tip}/\pi R^2$.

Equation 2.8 implies that $p(x, L)$ is L-independent, namely

$$p(x) \;=\; (f_{\rm tip}/\pi R^2)\exp[(2\alpha/R)x]\;; \tag{2.9}$$

Fig. 2.2 plots $p(x)$ for the flagellum. Equation 2.9, combined with Eq. 2.3, now yields an expression for friction density,

$$g(x) \;=\; (f_{\rm tip}\alpha/\pi R^2)\exp[(2\alpha/R)x]\;. \tag{2.10}$$

A crucial property controlling translocation in the flagellar channel is the isothermal compressibility, $\kappa_{\rm T}$, of the translocated flagellin. For a given relationship between exerted pressure and flagellin density, the compressibility is $\kappa_{\rm T} = -V^{-1}(dV/dp)_{\rm T}$ where $V = N\,\rho^{-1}$. If the flagellin atoms were disconnected, non-interacting point particles, the ideal gas law would hold, namely, $p = k_{\rm B}\,T\,\rho$. This pressure-density relationship is clearly an oversimplification. A more realistic description represents flagellin as a so-called Gaussian chain [77, 78]. In the case of this simple polymer model holds approximately the relationship

$$p \;\approx\; k_{\rm B}T\left[\tilde{\rho} \;+\; \frac{1}{3}b\,\pi^4 R^4 L_0\,\tilde{\rho}^3\right] \tag{2.11}$$

as we demonstrate in Appendix A. Here b is the so-called effective bond length for a Gaussian chain (b equals two times the polymer persistence length $l_{\rm p}$ [78]), L_0 the length of the polymer chain when it is

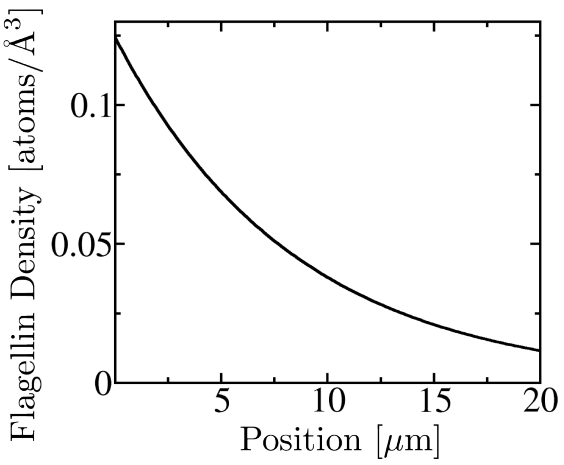

Figure 2.3: Flagellin Density. Using the system parameters listed in Table 2.2, including P = 100 (kcal/mol)/s, with Eq. 2.13 of the main text, flagellin density is shown as a function of position along the flagellum for the case $L = 20$ μm. Density is highest at the flagellum base (position = 0) and lowest at the flagellum tip (position = 20 μm). This plot reveals the flagellin density range relevant to the flagellum, namely density in the range [0.011, 0.12] atoms/Å^3. For comparison, the density of fully stretched flagellin in the channel is ∼0.01 atoms/Å^3, while the density of bulk water is ∼0.1 atoms/Å^3.

totally stretched, and $\tilde{\rho}$ the density of one polymer chain, defined through the polymer atom density / number of the atoms in a single polymer. One can see from Eq. 2.11 and Appendix A that for increasing $\tilde{\rho}$ holds $p \approx \gamma\,\tilde{\rho}^{\beta}$ with $\beta = 3$. In this case holds $\kappa_{\rm T} = 1/\beta p$. In a more realistic description the actual exponent β should differ from $\beta = 3$ due to atomic interactions that are neglected in the Gaussian chain model [77]. Accordingly, we stipulate

Condition 5 *Flagellin pressure is a power of flagellin density,*

namely,

$$p \;=\; \gamma\,(\rho/\rho_0)^{\beta} \tag{2.12}$$

where ρ_0 =1 atom/Å^3 is a reference pressure.

According to Eq. 2.9 and Eq. 2.12, flagellin density can now be expressed

$$\rho(x) = (f_{\rm tip}/\gamma\pi R^2)^{1/\beta}\exp\left[2\alpha\; x/R\beta\right]\;\rho_0\;; \tag{2.13}$$

Fig. 2.3 plots $\rho(x)$ for the flagellum. The flagellin flux, ϕ, is the product of flagellin density, ρ, and translocation velocity, v,

$$\phi(x,L) \;=\; \rho(x)\,v(x,L)\;. \tag{2.14}$$

For the density of flagellin in the channel to be stationary (time-independent), protein flux must be inde-

pendent of x. Equation 2.14, therefore, simplifies to

$$\phi(L) \;=\; \rho(x)v(x,L)\;. \tag{2.15}$$

This expression is most readily evaluated at $x = L$; for the evaluation we assume

Condition 6 *Flagellin monomers are pumped into the channel at a constant power, P.*

The pump power, P, equals the product of pressure, cross sectional area and translocation velocity at $x = L$, i.e., $P = \pi R^2 p(x)\, v(x,L)\big|_{x=L}$, which can be written

$$v(x,L)|_{x=L} \;=\; P/\left[\pi R^2 p(x)\right]\big|_{x=L}\;. \tag{2.16}$$

Equation 2.15, combined with $v(x,L)|_{x=L}$ from Eq. 2.16, $\rho(x)|_{x=L}$ from Eq. 2.13 and $p(x)|_{x=L}$ from Eq. 2.9, becomes

$$\phi(L) \;=\; \phi_0 \exp\left[(2\alpha/R)\,(1/\beta \,-\, 1)\,L\right] \tag{2.17}$$

where

$$\phi_0 \;=\; \rho_0\;(P/f_{\rm tip})\left(f_{\rm tip}/\gamma\pi R^2\right)^{1/\beta}\;. \tag{2.18}$$

Finally, one can substitute in Eq. 2.15 $\phi(L)$ from Eq. 2.17 and $\rho(x)$ from Eq. 2.13 and obtain

$$v(x,L) \;=\; (P/f_{\rm tip}) \exp\left\{-\,(2\alpha/R)\left[L \,-\, (L \,-\, x)\,/\beta\right]\right\}\;; \tag{2.19}$$

Fig. 2.4 shows $v(x,L)$ for the flagellum.

2.2.2 Rate of Flagellin Translocation and Flagellum Growth

We can now express the growth velocity of the flagellum, $V(L)$ (in units length/time), as the product of the frequency with which flagellin monomers reach the tip, $\pi R^2 \phi(L)$ and the length which each flagellin contributes to L, $\lambda = 5$ Å/flagellin, namely

$$V(L) \;=\; \lambda\,\pi\,R^2\,\phi(L)\;, \tag{2.20}$$

which, by substituting $\phi(L)$ from Eq. 2.17, becomes

$$V(L) \;=\; V_0 \exp\left(-\,L/a\right) \tag{2.21}$$

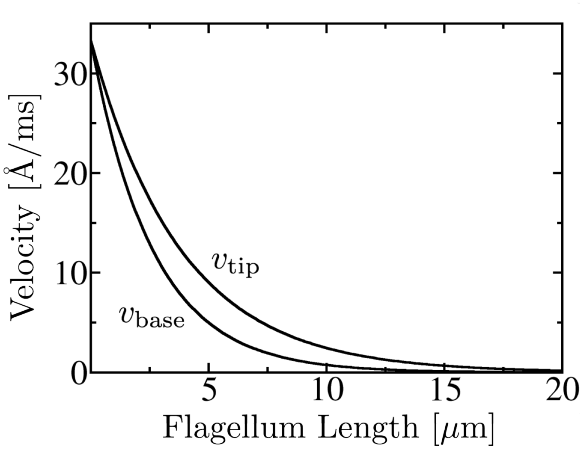

Figure 2.4: Flagellin Translocation Velocity. Using the system parameters listed in Table 2.2, including $P = 100$ (kcal/mol)/s, with Eq. 2.19 of the main text, flagellin translocation velocity is shown as a function of flagellum length. The upper curve plots $v_{\rm tip} = v(x = 0, L)$, the translocation velocity at the flagellum tip, where velocity is always highest; the lower curve plots $v_{\rm base} = v(x = L, L)$, the translocation velocity at the flagellum base, where velocity is always lowest. For $L = 0$ (nascent flagellum) tip and base translocation velocity are equal and highest since there is very little friction built up yet in the flagellum. As the flagellum grows to its maximal length, both $v_{\rm tip}$ and $v_{\rm base}$ decay to zero. The area between the two curves bounds the range of relevant translocation velocities as a function of flagellum length.

where

$$a \;=\; (R/2\alpha)\,/\,(1 \,-\, 1/\beta) \tag{2.22}$$

and

$$V_0 \;=\; \lambda\,\rho_0\,\left(P\pi R^2/f_{\rm tip}\right)\left(f_{\rm tip}/\gamma\pi R^2\right)^{1/\beta}\;. \tag{2.23}$$

For $\beta \;\rightarrow\; 1$ the decay length, a, becomes infinite and, therefore, the growth rate would be constant such that $V(L) = V_0$ holds in contrast to observation [27]; thus it is essential that $\beta > 1$ be established.

By defining the growth rate as the change in length with respect to time, i.e., $V = dL(t)/dt$, Eq. 2.21 can be integrated to express flagellum length as a function of time,

$$dL(t)/dt \;=\; V_0 \exp\left[-\,L(t)/a\right] \tag{2.24}$$

the solution of which, for $L(0) = 0$, is

$$L(t) \;=\; a\,\ln\left(1 \,+\, V_0\,t/a\right)\;. \tag{2.25}$$

Fig. 2.10 presents $L(t)$ for the flagellum.

This concludes the derivation of expressions for length, growth rate and the five flagellin translocation

properties. These expressions offer a wealth of insight into flagellin translocation through the long, confining flagellar channel and why the growth rate decays exponentially with length. Equation 2.13 states that, as the flagellum grows, flagellin density at the base increases. As a result, friction at the base grows (c.f., Eq. 2.10). Increasing friction causes pressure to increase at the base (c.f., Eq. 2.9). Since the secretion system operates at constant power, the increasing pressure required to pump in flagellin at the base means that the rate of insertion must decrease proportionally, thereby slowing the rate of translocation and, consequently, filament growth.

According to Eq. 2.25 the flagellum can grow infinitely long. What then, halts the flagellum's growth at $L_{\max} = 20$ μm [27]? The answer is that even though the pump at the base of the filament operates at a constant power P, it's force cannot increase indefinitely. In fact, when the pump reaches it's stall force, f_{stall}, filament elongation ceases, namely (c.f. Eq. 2.9), for

$$f_{\text{stall}} \;=\; f_{\text{tip}} \exp\left(2\alpha L_{\max}/R\right) \;. \tag{2.26}$$

From this follows

$$L_{\max} \;=\; (R/2\alpha) \ln(f_{\text{stall}}/f_{\text{tip}}) \;. \tag{2.27}$$

The result shows that the maximum length of a flagellum can be extended by either increasing the ratio $f_{\text{stall}}/f_{\text{tip}}$, or by decreasing the coefficient of friction, α, defined in Eq. 2.3. Due to the weak, namely only logarithmic, dependence on $f_{\text{stall}}/f_{\text{tip}}$, α is the primary control of $L_{\max}$.

2.2.3 Verification of Flagellin Translocation Model

Below, Conditions 1 and 5 are verified for the flagellum through molecular dynamics simulations. It will be demonstrated how compressing flagellin in a cylinder allows one to measure the needed physical characteristics as a function of flagellin density. Conditions 2 and 3 are assumed to hold true for the flagellum since the translocation velocity is extremely slow compared to the relaxation processes during translocation. Condition 2 relies on theoretical arguments given above, the key one being a linear relationship between pressure and the average number of adhesion points as stated by Eq. 2.2. Certainly, the linearity stated in Eq. 2.3 can hold only over a limited pressure interval, but should hold for the low pressures as they arise in flagellar transport. This will be demonstrated below. Condition 4 relates to the flagellum cap protein, which helps the flagellin fold into place at the filament tip. Condition 6 is evident for biological pumps, which operate by ATP hydrolysis at some fixed rate and have been shown to slow down under increasing load [79, 80].

2.2.4 Molecular Dynamics Simulation Methods

Ten non-equilibrium all-atom molecular dynamics simulations were performed (see Table 2.1) to study flagellin translocation and verify the conditions underlying our theoretical description. The first nine simulations involve a segment of unfolded flagellin being compressed in a cylinder (see Fig. 2.5); the tenth simulation is for a segment of flagellin translocated through the actual flagellar channel (see Fig. 2.6). We carried out also 60 simulations of a flagellin segment equilibrating in implicit solvent while confined to cylinders of different, but fixed, lengths and radii. Simulations were prepared and analyzed using VMD [81] and carried out with NAMD [41]. In each simulation, temperature was held at 300 K by a Langevin thermostat, and a pressure of 1 atm was maintained by a Nosé-Hoover Langevin-piston barostat with a period of 1 ns and a decay rate of 300 ps; periodic boundary conditions were assumed. Multiple time stepping was employed using an integration timestep of 1 fs, with short-range forces evaluated every two timesteps and long-range electrostatic forces every four timesteps. The short-range forces were smoothed with a cutoff between 10 and 12 Å, while long-range electrostatic interactions were calculated using the particle-mesh Ewald algorithm. All simulations used the CHARMM22 [82] force field together with the TIP3P water model [83]. A salt strength of 200 mM KCl was assumed to neutralize the charge of the system and in keeping with experiment [27]. The structure for flagellin monomers of *Salmonella typhimurium* was obtained from the *Protein Data Bank*; PDB code 1UCU [28]. The structure of the flagellar filament was solved by Yonekura et al. [25, 28] and others [66, 67, 84].

Name	Type	Residues	Channel	Atoms	Time
$5A$	Comp	1-164	5 Å cyl	23,232	10 ns
$5B$	Comp	165-329	5 Å cyl	23,142	10 ns
$5C$	Comp	330-494	5 Å cyl	24,988	10 ns
$7A$	Comp	1-164	7 Å cyl	31,811	10 ns
$7B$	Comp	165-329	7 Å cyl	32,721	10 ns
$7C$	Comp	330-494	7 Å cyl	33,800	10 ns
$9A$	Comp	1-164	9 Å cyl	34,186	10 ns
$9B$	Comp	165-329	9 Å cyl	32,077	10 ns
$9C$	Comp	330-494	9 Å cyl	31,787	10 ns
CD0	Trans	1-100	CD0	69,664	52 ns
Equil	Equil	1-164	60 cyls	2,445	447 ns

Table 2.1: Simulations performed. The first nine are simulations of unfolded flagellin being compressed in a cylinder. The tenth simulation models flagellin translocation through the flagellum channel. The Equil group of 60 simulations equilibrate flagellin segment A in cylinders of different geometries; simulations employed an implicit solvent description.

Figure 2.5: Flagellin compression simulation. To study the effects of confinement and density on translocation, three denatured flagellin segments, A, B and C, were compressed in cylinders of radii 5, 7 and 9 Å. The piston walls, shown as black vertical lines, were moved toward each other from time t_0 (flagellin backbone in blue) to time t_f (flagellin backbone in red) at a velocity of 10 Å/ns. Arrows indicate the piston force.

2.2.5 Flagellin Preparation

Flagellin was unfolded by fixing the N-terminal residue and pulling the C-terminal residue by steered molecular dynamics [85] in vacuum to a linear density of 0.25 res/Å. Once stretched, the monomer was divided into three segments of equal length for the purpose of accelerated independent sampling: segment A with residues 1-164, segment B with residues 165-329, and segment C with residues 330-494. Copies of each stretched segment contracted and then equilibrated for 3 ns in a periodic water box while confined to cylinders of three different radii, namely 5, 7 and 9 Å, making nine segments, labeled 5A through 9C, as listed in Table 2.1. Confinement to cylinders, achieved computationally through a Tcl force [41] (an external, user-specified force), involved a radial force $f = -k \times \max(r - r_c, 0)$ to the backbone atoms; r is the atom's distance from the cylinder axis, r_c is the cylinder radius; for k we assumed a value 10 (kcal/mol)/Å^2.

2.2.6 Compression

The flagellin segments, still confined to their respective radii, were axially compressed by another Tcl force [41], on the right side by $f = -k \times \max(x - x_c, 0)$, and on the left side by $f = +k \times \min(x + x_c, 0)$, where x is the atom's x-coordinate (along the axis of the cylinder) and $\pm x_c$ is the position of the right and left compression walls (shown as vertical lines in Fig. 2.5). The walls move inward as the compression progresses according to $x_c = x_0 - vt$ where $x_0 = 120$ Å is the initial position of the compression walls, $v = 10$ Å/ns is the velocity at which the compression walls move inward, and t is the simulation time; one simulated system is illustrated in Fig. 2.5. Axial and radial pressure were calculated as the force exerted by the cylinder's two axial walls and radial wall on the protein, respectively, divided by their respective areas. Density was calculated as the number of protein atoms divided by the cylinder's changing volume.

2.2.7 Equilibration

To test the relationship between axial and radial pressure and density at equilibrium, flagellin segment A was equilibrated in cylinders of several lengths (L=150, 165, 180, 195, 210, 225 Å) and radii (R = 5, 6, 7, 8, 9, 10, 11, 12, 13, 14 Å) for several nanoseconds each; larger cylinders were equilibrated longer according to radius: 5 Å for 5.5 ns, 6-7 Å for 6 ns, 8-9 Å for 7 ns, 10-11 Å for 8 ns and 12-14 Å for 9 ns, totaling 447 ns of equilibration time. For accelerated equilibration, the generalized Born implicit solvent, model $GB^{OBC}II$ [53], implemented in NAMD [41], was employed. The following parameters were changed in going from explicit to implicit solvent simulations: the barostat, periodic boundary conditions and PME were not employed; the nonbonded interaction cutoff was increased to 14 Å; the cutoff for calculating Born radii was set to 12 Å; an implicit ion concentration of 0.5 M was assumed. The surface contact density is measured as the number of surface contacts (defined as any atom outside the cylinder) divided by the cylinder radial surface averaged over the trajectory.

2.2.8 Translocation

Flagellin translocation was simulated by pushing a denatured segment of flagellin through the flagellar channel. The initial flagellar channel was built from 44 repeating flagellin monomers arranged helically according to the flagellum structure reported in [25]. The flagellin segment resulting from simulation 5A (see Table 2.1), after equilibration, but before compression, was truncated to 100 residues in order to fit the flagellum segment. This truncated segment still represents the behavior of the entire flagellin as a denatured protein, and contains the charged residues which are important for simulating the transient salt bridges. The flagellin was placed in the filament channel without cylindrical restraints and, after removing overlapping water molecules, the full system was equilibrated for 1 ns. During translocation, the backbone atoms of the channel were held in place by harmonic restraints ($k = 1$ (kcal/mol)/Å), while a Tcl force [41] was applied to the flagellin's backbone atoms to induce translocation. The force applied to each backbone atom was $f = -k \times \min(x - x_0, 0)$, with $k = 1$ (kcal/mol)/Å; x is the atom's coordinate along the channel axis, and x_0 the wall position; the wall was moved toward the tip at a velocity of 100 Å/ns for 2.5 ns and then at a velocity of 1 Å/ns for 52 ns. The simulated channel originally required four full repeats (44 monomers) to span the truncated flagellin segment 5A; after 2.5 ns, the 5A segment had compressed to half it's original length such that the channel could be shortened to two repeats without affecting system behavior. Because the CD0 helices are the only part of the flagellar channel in contact with the translocating flagellin [28], all other domains were removed from the channel structure, reducing computational cost; the resulting CD0 channel with translocating flagellin is shown in Fig. 2.6.

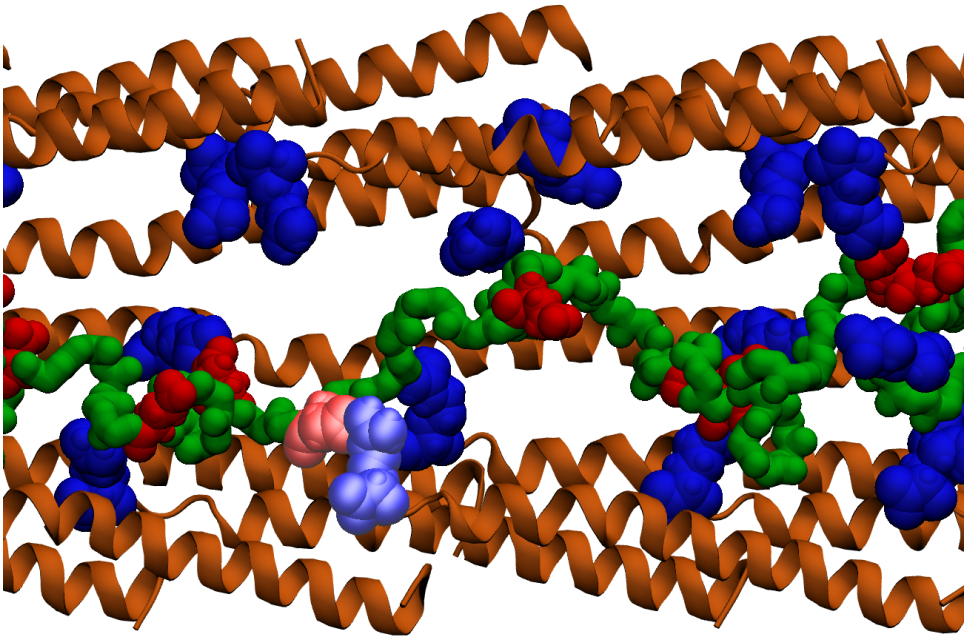

Figure 2.6: Flagellin translocation simulation. To simulate flagellin translocation through the flagellum, flagellin was pushed 52 Å through a channel lined by the CD0 domain (light brown). The repeating arginine residues of the CD0 domains are shown in dark blue; the acidic and basic residues of the translocated flagellin are shown in dark red and dark blue, respectively. One of several transient salt bridges is shown with the interacting acidic and basic residues colored light red and light blue, respectively.

2.3 Results

Molecular dynamics simulations carried out are analyzed to test Conditions 1, 2 and 5 and to illustrate the translocation of flagellin in the actual channel. It is crucial that our simulations probe flagellin properties in ranges relevant to the flagellum. For this reason, plots of $p(x)$, $\rho(x)$ and $v(x, L)$ are provided in Figs. 2.2, 2.3 and 2.4. Fig. 2.3 shows the relevant flagellin density range, $[0.011, 0.12]$ atoms/Å^3.

2.3.1 Test of Condition 1

Condition 1 states that the pressure of cylindrically confined, unfolded flagellin is isotropic. The nine compression and 60 equilibrium simulations allow complementary comparisons of axial and radial pressure. The compression simulations allow measurement of pressure across a continuous range of densities with the caveat that, even though the compression rate is slow, the compression process may still be off equilibrium. The equilibrium simulations, on the other hand, allow pressure measurements under equilibrium conditions, but only at discrete densities as determined by the selected volumes of the 60 cylinders. Fig. 2.7 presents the data for the compression simulations. The relationship between axial and radial pressure is $p_{\text{axial}} = 2.9\, p_{\text{radial}}$,

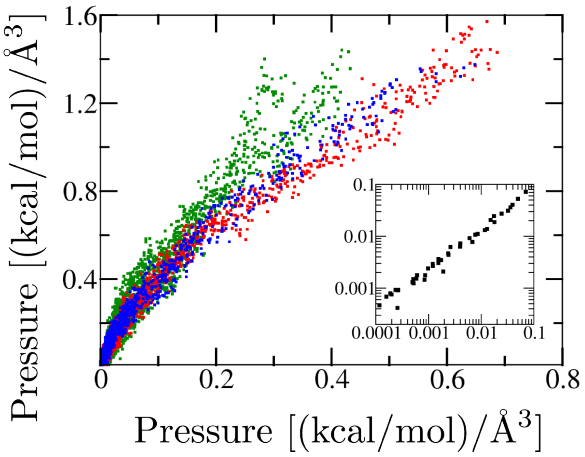

Figure 2.7: Isotropic pressure of unfolded flagellin. Radial (y-axis) versus axial (x-axis) pressure for nine compression simulations in cylinders of radius 5 Å (green), 7 Å (red) and 9 Å (blue). Each data point represents a 10 ps snapshot of the pressures as compression progresses (from bottom left to top right). Inset: Radial versus axial pressure for 60 equilibrium simulations; each point represents the average pressure of a single simulation. Linear least squares fit demonstrates $p_{\text{axial}} = 0.97\, p_{\text{radial}}$.

which verifies that axial and radial pressure are proportional under slow compression conditions. The axial pressure is larger than the radial pressure as the compression walls must move the flagellin tens of Å's through the viscous water that is getting expelled from the shrinking cylinder. Fig. 2.7 inset presents the data for the equilibrium simulations; under equilibrium conditions axial and radial pressure are observed to be related by $p_{\text{axial}} = 0.97\, p_{\text{radial}}$ which supports isotropicity and verifies Condition 1.

2.3.2 Test of Condition 2

The equilibrium simulations of confined flagellin allow comparison of the radial pressure of the flagellin and the degree to which it contacts the confining cylinder, representing the channel. Fig. 2.8, which plots surface contact density against radial pressure for the equilibrium simulations, shows that surface contact density for each independent simulation is directly proportional to radial flagellin pressure. This result verifies that increasing flagellin pressure increases contact density, which in turn increases static friction, as stipulated in Condition 2.

2.3.3 Test of Condition 5

Both compression and equilibrium simulations allow characterization of pressure and density for the confined unfolded flagellin. The pressure versus density data are shown in Fig. 2.9. To test Condition 5, the data presented in Fig. 2.9 were fitted to Eq. 2.12. For the compression simulations the axial and radial pressures (in units (kcal/mol)/Å^3) relate to density by $p_{\text{axial}} = 36\,(\rho/\rho_0)^3$ and $p_{\text{radial}} = 26\,(\rho/\rho_0)^{3.6}$ while the aggregate

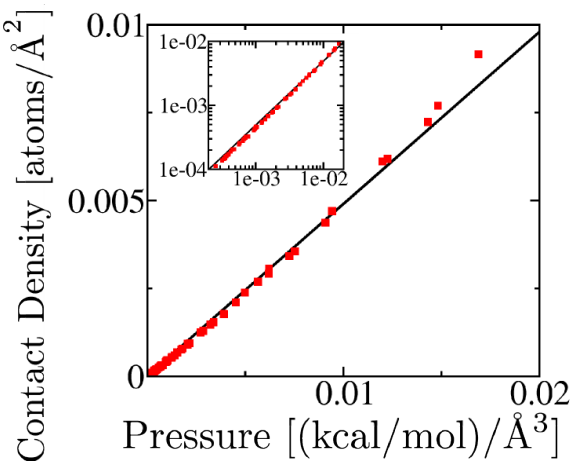

Figure 2.8: Surface Contact Density. Surface contact density, measured as number of atoms beyond the confining cylinder, versus flagellin pressure, measured as the total force of the confining cylinder on the flagellin segment, for each of the 60 equilibrium simulations. Samples at upper right are from simulations at higher density while samples at lower left are from simulations at lower density. Because atoms which contact the cylinder represent flagellin atoms which would interact with the flagellar channel (were it present), the density of contacts between unfolded flagellin and the confining channel are proportional to flagellin pressure, confirming Condition 2. Inset: The same data is presented in a log-log plot to highlight the proportional relationship across a wide range of flagellin densities.

data are described by $p = 29\,(\rho/\rho_0)^{3.1}$. Fig. 2.9 inset shows, for the equilibrium simulations, flagellin pressure and density related by $p = 15\,(\rho/\rho_0)^{3.2}$. While both types of simulation agree on $\beta \approx 3.2$, γ from the equilibrium simulations is half that of the compression simulations. The different γ values in equilibrium and non-equilibrium compression simulations arise from the flagellin in the compression simulation having to dynamically compress through viscous water and squeeze explicit water out of the shrinking cylinder, as it becomes increasingly occupied by compressed flagellin, which adds a compression rate-dependent pressure.

To illustrate in how far the Gaussian chain pressure - density relationship in Eq. 2.11 holds, we compare the prediction of this expression with the equilibrium simulation results (see data point in Fig. 2.9 denoted through an arrow) for the low density of $\rho_{\rm atom} = 3.07 \times 10^{-2}$ atoms/Å^3 and for $N_{\rm atom} = 2445$ (i.e., $\tilde{\rho} = \rho_{\rm atom}/N_{\rm atom} = 1.25 \times 10^{-5}$ Å^{-3}) as well as for $R = 13$ Å, $L = 150$ Å, $L_0 \approx 164 \times 5$ Å, and $T = 300$ K. For a choice $b = 40.5$ Å, one obtains from Eq. (2.11) agreement with the simulated pressure $p = 4.33 \times 10^{-5}$ kcal/(mol Å^3). The persistence length of the flexible polymer chain would be half of the effective bond length, b [78], i.e., would be 20 Å, which falls into the persistence length range of 5-25 Å expected for unfolded protein [86–88]; for example, the observed and theoretical persistence length of poly-L-alanine is about 20 Å (see [89], Chap. 18).

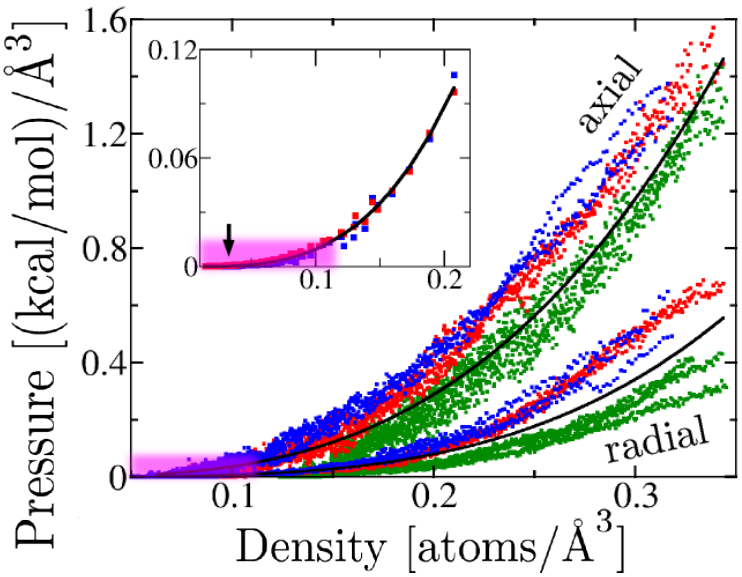

Figure 2.9: Pressure versus density of unfolded flagellin. Flagellin pressure versus density for nine compression simulations at cylinder radius 5 Å (green), 7 Å (red) and 9 Å (blue). The upper three curves show axial pressure while the lower three show radial pressure. Each data point represents a 10 ps snapshot as the compression progresses (from bottom left to top right). The two black curves represent the average fit of Eq. 2.12 to the axial ($p = 36\ (\rho/\rho_0)^3$) and radial ($p = 26\ (\rho/\rho_0)^{3.6}$) pressures. Inset: Radial (red) and axial (blue) pressure versus density for 60 equilibrium simulations (each data point marks the average pressure and density of a single simulation). The black curve represents the average fit of Eq. 2.12 to the axial and radial pressures ($p = 15\ (\rho/\rho_0)^{3.2}$). The arrow indicates location of a $R = 13$ Å, $L = 150$ Å data point used to match the Gaussian chain model to simulation data, using b (c.f. Eq. 2.11) as the fitting parameter. Purple highlight is added to illustrate the pressure and density range relevant to the flagellum.

2.3.4 Flagellin Translocation

To view the behavior of translocating flagellin in it's channel environment, a 100-residue segment of flagellin was pushed through a flagellar channel as illustrated in Fig. 2.6. During the translocation simulation, of the many flagellin-channel interactions observed, the most striking ones are a series of salt bridges formed; a salt bridge was considered to be formed if the oxygen atom of the acidic residue and the nitrogen atom of the basic residue came within a distance of 3.2 Å. During the 52 ns simulation, 30 salt bridges were observed, 18 within flagellin and 12 between flagellin and channel. The flagellin-channel salt bridges persisted, on average, for 1-1.5 ns, corresponding to 1-1.5 Å displacement of the flagellin along the channel; at the much slower *in vivo* translocation rate, these salt bridges may persist much longer.

The prevalence of salt bridges stimulated the investigation of the importance of charged residues across different species of flagellin. Using the VMD [81] plugin MultiSeq [90], multiple sequence alignment of both *Salmonella* and E. coli flagellin showed a conserved SLLX motif at the C-terminus (the terminus exposed to the channel interior). In *Salmonella*, X is ARG494, which places a positively charged residue along the inner surface of the channel. In E. coli, X changes to GlnArg, which eliminates this positive charge; however, there is a co-mutation of GLN481 to LYS481; LYS481 is spatially adjacent to X and, thus, restores the positively charged residue to location 494. The conservation of a charged residue X suggests it to be important for translocation. Studies of the flagellum with X mutated to neutral residues could shed light on the impact of salt bridge formation on translocation.

2.3.5 Agreement with Experiment

It is remarkable that our simple model of flagellin translocation agrees so well with available experimental data on how friction in the channel causes the growth rate to decay exponentially with length. The strength of our model is further demonstrated by comparison with experiment.

For quantitative study of the flagellum, we first identify the physical system parameters needed in our description; Table 2.2 presents the required parameters. $L_{\max}$, V_0, ϕ_0, a and λ are experimentally known for the flagellum, while β and γ are taken from the fit of Eq. 2.12 to the equilibrium simulations as presented in Fig. 2.9. The power of the flagellar pump, P, (from which f_{tip} and f_{stall} can be directly calculated) is not experimentally known; we therefore list in Table 2.2 reasonable values for P and the respective values for f_{stall} and f_{tip}. For quantitative analysis we will assume $P = 100$ (kcal/mol)/s, taken from the viral DNA packaging motor [79, 80]. Naturally, following our description and simulations, the system parameters listed in Table 2.2 can be determined for flagella of other species.

The parameter α is neither known experimentally nor can it be determined directly for our simulations;

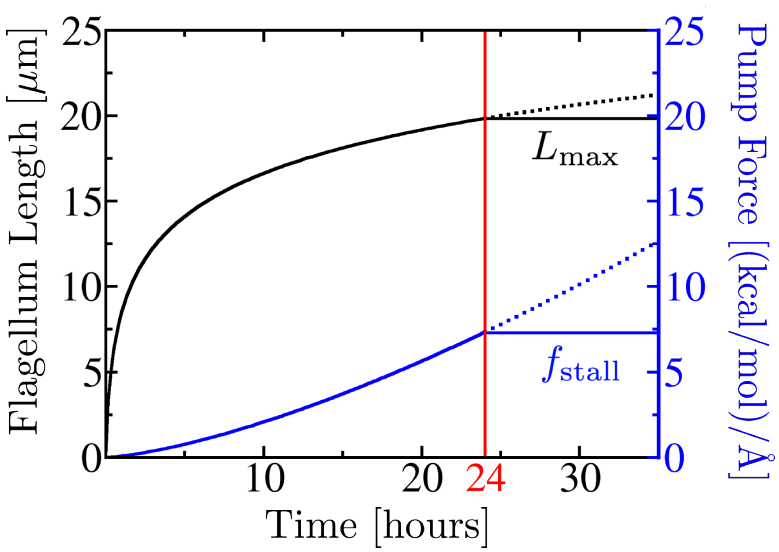

Figure 2.10: Flagellin Growth. Using the system parameters listed in Table 2.2, including $P = 100$ (kcal/mol)/s, flagellum length (black, c.f. Eq. 2.25) and pump force (blue, c.f. Eq. 2.9) is shown as a function of time. As the flagellum grows, the force required to pump additional flagellin into the channel increases (due to increasing $p(x = L)$). At $t = 24$ hrs (red) the pump force reaches the stall force, $f_{\text{stall}} = 7$ (kcal/mol)/Å, and the pump can not surmount the force required to pump additional flagellin into the channel (dotted blue). At this point, the flagellum can not grow longer (black dotted); instead it remains at its maximal length, $L_{\max} = 20$ μm.

α can, however, be computed by employing known parameters a, R, and β in Eq. 2.22. The very small $\alpha = 2 \times 10^{-4}$, pivotal to the relationship between pressure and friction (c.f. Eq. 2.3), suggests that very little of the radial force is translated into axial static friction, consistent with the low forces of creeping static friction. In contrast, an α value of $\sim$1, would decrease the decay length from $a = 3.6$ μm to $a = 3.6$ Å, not allowing the flagellum to grow.

From Eq. 2.25, it can be calculated that the flagellum grows to its maximal length in roughly 24 hrs; this growth time is consistent with [27] where it is reported that flagella grow to ~ 10 μm in only a few hours, but require more than 10 hrs to reach 15 μm. We argued above (c.f. Eq. 2.27) that the pump's stall force prevents the flagellum from infinite growth. Employing the parameters of Table 2.2 in Eq. 2.27 returns that a stall force of 7 (kcal/mol)/Å corresponds to a maximal length of 20 μm; this stall force agrees qualitatively with the viral DNA packaging motor's stall force of $\sim$1 (kcal/mol)/Å [79]. Fig. 2.10 illustrates how the force at the base of the flagellum, required to pump additional flagellin monomers into channel, grows with time; when the force at the flagellum base reaches f_{stall}, growth ceases.

Param	Value	Source
(1) $L_{\max}$	20 μm	Ref. [27]
(2) V_0	91 Å/s	Ref. [27]
(3) ϕ_0	435 atoms/Å^2/s	Ref. [27]
(4) a	3.7 μm	Ref. [27]
(5) λ	5 Å/7197 atoms	Ref. [65]
(6) R	10 Å	Ref. [28]
(7) β	3.2	Fig. 2.9
(8) γ	15 (kcal/mol)/Å^3	Fig. 2.9
(9) α	2×10^{-4}	Eq. 2.22

P	f_{tip}	f_{stall}
(a) 1	3e-6	8e-3
(b) 10	9e-5	0.2
(c) 100	3e-3	7

Table 2.2: Flagellin translocation parameters. Parameters 1-6 are taken from the listed references; parameters 7-8 are measured from the equilibrium simulations; parameter 9 is calculated from the previously listed values and the listed equation. Regarding parameter 5, each flagellin monomer consists of 7197 atoms and contributes roughly 5 Å to flagellum length. Items a-c list three possible P (in units (kcal/mol)/s) values along with the respective f_{tip} and f_{stall} (in units (kcal/mol)/Å) values that are consistent with Eq. 2.26 and Eq. 2.27.

2.4 Summary

The exploration of flagellin translocation was facilitated by guidance of a theoretical model, which narrows the scope of properties which simulations need to address. For example, though the theoretical model describes several flagellin properties as a function of x, it also allows the properties to be expressed in terms of density independent of x; therefore, the simulations could explore friction and pressure as a function of density instead of having to simulate along an extremely long filament focusing on x-dependence. The theoretical model also furnished clear tests for validation as well as a clear framework for interpreting the results in terms of actual flagellin properties.

Our model of flagellin translocation links local and short time properties to overall translocation, allowing the compression simulations, which cover only nanometers in size and nanoseconds in time, to represent the growing flagellum, which grows several micrometers in length over several hours.

The theoretical description concludes that the flagellum growth rate decreases exponentially with length because of protein compression and friction between translocating flagellin and flagellar channel increasing proportionally along the flagellum. The growth rate calculated agrees with Iino [27] who experimentally measured the growth rate decaying exponentially with length and showed it to be caused by decreasing translocation efficiency.

Chapter 3

Implicit Solvent

3.1 Introduction

3.1.1 Solvent in Molecular Dynamics

Molecular dynamics (MD) is a computational method [1] employed for studying the dynamics of nanoscale biological systems on nanosecond to microsecond timescales [2]. Using MD, researchers can utilize experimental data from crystallography and cryo-electron microscopy (cryo-EM) to explore the functional dynamics of biological systems [3].

Because biological processes take place in the aqueous environment of the cell, a critical component of any biological MD simulation is the solvent model employed [34, 35]. An accurate solvent model must reproduce water's effect on solutes such as the free energy of solvation, dielectric screening of solute electrostatic interactions, hydrogen bonding and van der Waals interactions with solute. For typical biological MD simulations, solute is comprised of proteins, nucleic acids, lipids or other small molecules.

Two main categories of solvent models are explicit and implicit solvents. Explicit solvents, such as SPC [91] and TIP3P [83], represent water molecules explicitly as a collection of charged interacting atoms and calculate a simple potential function, such as Coulomb electrostatics, between solvent and solute atoms. Implicit solvent models, instead, ignore atomic details of solvent and represent the presence of water indirectly through complex interatomic potentials between solute atoms only [52, 92, 93]. There are advantages and disadvantages of each solvent model.

Simulation of explicit water is both accurate and natural for MD, but often computationally too demanding, not only since the inclusion of explicit water atoms increases a simulation's computational cost through the higher atom count, but also because water slows down association and disassociation processes

Material in this chapter is reproduced in part with permission from David E. Tanner, Kwok-Yan Chan, James C. Phillips and Klaus Schulten, "Parallel generalized Born implicit solvent calculations with NAMD," *Journal of Chemical Theory and Computation*, 7:3635-3642 (2011).

due to the relatively long relaxation times of interstitial water [94]. The viscous drag of explicit water also retards large conformation changes of macromolecules [95].

3.1.2 Implicit Solvent

An alternative representation of water is furnished by implicit solvent descriptions which eliminate the need for explicit solvent molecules. Implicit water remains always equilibrated to the solute. The absence of explicit water molecules also eliminates the viscosity imposed on simulated solutes, allowing faster equilibration of solute conformations and better conformational sampling. Examples of popular implicit solvent models are Poisson-Boltzmann electrostatics [96, 97], screened Coulomb potential [92, 98], analytical continuum electrostatics [99] and generalized Born implicit solvent [100].

The generalized Born implicit solvent (GBIS) model, used by MD programs CHARMM [36, 37], Gromacs [38, 39], Amber [40] and NAMD [41, 42], furnishes a fast approximation for calculating the electrostatic interaction between atoms in a dielectric environment described by the Poisson-Boltzmann equation. The GBIS electrostatics calculation determines first the Born radius of each atom, which quantifies an atom's exposure to solvent, and, therefore, its dielectric screening from other atoms. The solvent exposure represented by Born radii can be calculated with varying speeds and accuracies [101] either by integration over the molecule's interior volume [102, 103] or by pairwise overlap of atomic surface areas [100]. GBIS calculations then determine the electrostatic interaction between atoms based on their separation and Born radii.

GBIS has benefited MD simulations of small molecules [104]. For the case of large systems, whose large conformational motions [105] may benefit most from an implicit solvent description, but which must be simulated on large parallel computers [106], the challenge to develop efficient parallel GBIS algorithms remains. In the following, we outline how NAMD addresses the computational challenges of parallel GBIS calculations and efficiently simulates large systems, demonstrated through benchmarks as well as simulations of the *Escherichia coli* ribosome, a RNA-protein complex involving ∼250,000 atoms.

3.2 Methods

In order to characterize the challenges of parallel generalized Born implicit solvent (GBIS) simulations, we first introduce the key equations employed. We then outline the specific challenges that GBIS calculations pose for efficient parallel performance as well as how NAMD's implementation of GBIS achieves highly efficient parallel performance. GBIS benchmark simulations, which demonstrate NAMD's performance, as well as the ribosome simulations, are then described.

3.2.1 Generalized Born Implicit Solvent Model

The GBIS model [52] represents polar solvent as a dielectric continuum and, accordingly, screens electrostatic interactions between solute atoms. GBIS treats solute atoms as spheres of low protein dielectric ($\epsilon_{\rm p} = 1$), whose radius is the Bondi [107] van der Waals radius, in a continuum of high solvent dielectric ($\epsilon_{\rm s} = 80$).

The total electrostatic energy for atoms in a dielectric solvent is modeled as the sum of Coulomb and generalized Born (GB) energies [52],

$$E_{\rm T}^{\rm Elec} \;=\; E_{\rm T}^{\rm Coul} + E_{\rm T}^{\rm GB} \;. \tag{3.1}$$

The total Coulomb energy for the system of atoms is the sum over pairwise Coulomb energies,

$$E_{\rm T}^{\rm Coul} \;=\; \sum_i \sum_{j>i} E_{ij}^{\rm Coul} \;, \tag{3.2}$$

where the double summation represents all unique pairs of atoms within the interaction cutoff; the interaction cutoff for GBIS simulations is generally in the range 16-20 Å, i.e., longer than for explicit solvent simulations, where it is typically 8-12 Å. The reason for the wider cutoff is that particle-mesh Ewald summations, used to describe long-range Coulomb forces, cannot be employed for treatment of long-range GBIS electrostatics.

The pairwise Coulomb energy, $E_{ij}^{\rm Coul}$ in eq. 3.2, is

$$E_{ij}^{\rm Coul} \;=\; (k_{\rm e}/\epsilon_{\rm p})\, q_i q_j / r_{ij} \;, \tag{3.3}$$

where $k_{\rm e} = 332$ (kcal/mol)Å/e^2 is the Coulomb constant, q_i is the charge on atom i, and r_{ij} is the distance between atoms i and j. The total GB energy for the system of atoms is the sum over pairwise GB energies and self-energies given by the expression

$$E_{\rm T}^{\rm GB} \;=\; \underbrace{\sum_i \sum_{j>i} E_{ij}^{\rm GB}}_{\rm pair} + \underbrace{\sum_i E_{ii}^{\rm GB}}_{\rm self} \;, \tag{3.4}$$

where the pair-energies and self-energies are defined as [52]

$$E_{ij}^{\rm GB} \;=\; -\,(k_{\rm e} D_{ij})\, q_i q_j / f_{ij}^{\rm GB} \;. \tag{3.5}$$

Here, D_{ij} is the pairwise dielectric term [108], which contains the contribution of an implicit ion concentration

to the dielectric screening, and is expressed as

$$D_{ij} \;=\; (1/\epsilon_{\rm p}) - \exp\left(-\kappa f_{ij}^{\rm GB}\right)/\epsilon_{\rm s}\;, \tag{3.6}$$

where κ^{-1} is the Debye screening length which represents the length scale over which mobile solvent ions screen electrostatics. For an ion concentration of 0.2 M, room temperature water has a Debye screening length of $\kappa^{-1} = \sim$7 Å. $f_{ij}^{\rm GB}$ is [52]

$$f_{ij}^{\rm GB} \;=\; \sqrt{r_{ij}^2 + \alpha_i\alpha_j \exp\left(-r_{ij}^2/4\alpha_i\alpha_j\right)}\;. \tag{3.7}$$

The form of the pairwise GB energy in eq. 3.5 is similar to the form of the pairwise Coulomb energy in eq. 3.3, but is of opposite sign and replaces the $1/r_{ij}$ distance dependence by $1/f_{ij}^{\rm GB}$. The GB energy bears a negative sign because the electrostatic screening counteracts the Coulomb interaction. The use of $f_{ij}^{\rm GB}$, instead of r_{ij}, in eq. 3.5 heavily screens the electrostatic interaction between atoms which are either far apart or highly exposed to solvent. The more exposed an atom is to high solvent dielectric, the more it is screened electrostatically, represented by a smaller Born radius, α_i.

Accurately calculating the Born radius is central to a GBIS model as the use of perfect Born radii allows the GBIS model to reproduce, with high accuracy, the electrostatics and solvation energies described by the Poisson-Boltzmann equation [109], and does it much faster than a Poisson-Boltzmann or explicit solvent treatment [110]. Different GBIS models vary in how the Born radius is calculated; models seek to suggest computationally less expensive algorithms without undue sacrifice in accuracy. Many GBIS models [111] calculate the Born radius by assuming atoms are spheres whose radius is the Bondi [107] van der Waals radius and determine an atom's exposure to solute through the sum of overlapping surface areas with neighboring spheres [112]. The more recent GBIS model of Onufriev, Bashford and Case ($\rm GB^{OBC}$), applied successfully to MD of macromolecules [53, 113] and adopted in NAMD, calculates the Born radius as

$$\alpha_i \;=\; \left[(1/\rho_{i0}) - (1/\rho_i)\tanh\left(\delta\psi_i - \beta\psi_i^2 + \gamma\psi_i^3\right)\right]^{-1}\;, \tag{3.8}$$

where ψ_i, the sum of surface area overlap with neighboring spheres, is calculated through

$$\psi_i \;=\; \rho_{i0}\sum_j H(r_{ij},\rho_i,\rho_j)\;. \tag{3.9}$$

As explained in prior studies [53, 111, 112], $H(r_{ij},\rho_i,\rho_j)$ is the surface area overlap of two spheres based on

their relative separation, r_{ij}, and radii, ρ_i and ρ_{i0}; the parameters δ, β and γ in eq. 3.8 have been calculated to maximize agreement between Born radii described by eq. 3.8 and those derived from Poisson-Boltzmann electrostatics [53]. ρ_i and ρ_j are the Bondi [107] van der Waals radii of atoms i and j, respectively, while ρ_{i0} is the reduced radius, $\rho_{i0} = \rho_i - 0.09\,\text{Å}$, as required by GB^{OBC} [53].

The total electrostatic force acting on an atom is the sum of Coulomb and GB forces; the net Coulomb force on an atom is given by

$$\vec{F}_i^{\text{Coul}} \;=\; -\sum_j \left[dE_{\text{T}}^{\text{Coul}}/dr_{ij}\right] \hat{r}_{ij} \;. \tag{3.10}$$

whose derivative $(dE_{\text{T}}^{\text{Coul}}/dr_{ij})$ is inexpensive to calculate. The required derivatives $(dE_{\text{T}}^{\text{GB}}/dr_{ij})$ for the GB force, however, are much more expensive to calculate because E_{ij}^{GB} depends on interatomic distances, r_{ij}, both directly (c.f. eqs. 3.5 and 3.7) and indirectly through the Born radius (c.f. eqs. 3.5, 3.7, 3.8 and 3.9). The net GB force on an atom is given by

$$\begin{aligned}
\vec{F}_i^{\text{GB}} &= -\sum_j \left[dE_{\text{T}}^{\text{GB}}/dr_{ij}\right] \hat{r}_{ij} \\
&= -\sum_j \left[\sum_k \sum_{l>k} (\partial E_{\text{T}}^{\text{GB}}/\partial r_{kl})(dr_{kl}/dr_{ij}) + \sum_k (\partial E_{\text{T}}^{\text{GB}}/\partial \alpha_k)(d\alpha_k/dr_{ij})\right] \hat{r}_{ij} \\
&= -\sum_j \left[\partial E_{\text{T}}^{\text{GB}}/\partial r_{ij} + (\partial E_{\text{T}}^{\text{GB}}/\partial \alpha_i)(d\alpha_i/dr_{ij}) + (\partial E_{\text{T}}^{\text{GB}}/\partial \alpha_j)(d\alpha_j/dr_{ij})\right] \hat{r}_{ij} \;,
\end{aligned} \tag{3.11}$$

with $\vec{r}_{ij} = \vec{r}_j - \vec{r}_i$. The required partial derivative of E_{T}^{GB} with respect to a Born radius, α_k, is

$$\partial E_{\text{T}}^{\text{GB}}/\partial \alpha_k = \sum_i \sum_{j>i} \left[\partial E_{ik}^{\text{GB}}/\partial \alpha_k + \partial E_{kj}^{\text{GB}}/\partial \alpha_k\right] + \sum_i \partial E_{ii}^{\text{GB}}/\partial \alpha_k \;. \tag{3.12}$$

The summations in eqs. 3.9, 3.11 and 3.12, require three successive iterations over all pairs of atoms for each GBIS force calculation, whereas calculating Coulomb forces for an explicit solvent simulation requires only one such iteration over atom-pairs. Also, because of the computational complexity of the above GBIS equations, the total cost of calculating the pairwise GBIS force between pairs of atoms is $\sim 7\times$ higher than the cost for the pairwise Coulomb force. For large systems and long cutoffs, the computational expense of implicit solvent simulations can exceed that of explicit solvent simulations; however, in this case, an effective speed-up over explicit solvent still arises due to faster conformational exploration as will be illustrated below. The trade-off between the speed-up of implicit solvent models and the higher accuracy of explicit solvent models is still under investigation [110]. Differences between GBIS and Coulomb force calculations create challenges for parallel GBIS simulations that do not arise in explicit solvent simulations.

3.2.2 Challenges in Parallel Calculation of GBIS Forces

Running a MD simulation in parallel requires a scheme to decompose the simulation calculation into independent work units that can be executed simultaneously on parallel processors; the scheme employed for decomposition strongly determines how many processors the simulation can efficiently utilize and, therefore, how fast the simulation will be. For example, a common decomposition scheme, known as spatial or domain decomposition, divides the simulated system into a three-dimensional grid of spatial domains whose side length is the interaction cutoff distance. Because the atoms within each spatial domain are simulated on a single processor, the number of processors utilized equals the number of domains.

Although explicit solvent MD simulations perform efficiently in parallel, even for simple schemes such as naive domain decomposition, the GBIS model poses unique challenges for simulating large systems on parallel computers. We outline here the three challenges arising in parallel GBIS calculations and how NAMD addresses them. For the sake of concreteness, we use the SEp22 dodecamer (PDB ID 3AK8) as an example as shown in Fig. 3.1.

3.2.3 *Challenge 1: Dividing workload among processors*

With a 12 Å cutoff, traditional domain decomposition divides the SEp22 dodecamer explicit solvent simulation (190,000 protein and solvent atoms) into $7 \times 7 \times 7 = 343$ domains which efficiently utilize 343 processors (see Fig. 3.1A). Unfortunately, with a 16 Å cutoff for the implicit solvent treatment, the same decomposition scheme divides the system (30,000 protein atoms) into $4 \times 4 \times 4 = 64$ domains which can only utilize 64 processors (see Fig. 3.1B). An efficient parallel GBIS implementation must employ a decomposition scheme which can divide the computational work among many (hundreds or thousands) processors (see Fig. 3.1C).

3.2.4 *Challenge 2: Workload imbalance from spatially heterogenous atom distribution*

Due to the lack of explicit water atoms, the spatial distribution of atoms in a GBIS simulation (see Fig. 3.2) is not uniform as it is for an explicit solvent simulation (see Fig. 3.1A); some domains contain densely packed atoms while others are empty (see Fig. 3.1B). Because the number of atoms varies highly among implicit solvent domains, the workload assigned to each processor also varies highly. The highly varying workload among processors for domain decomposition causes the naive decomposition scheme to be inefficient and, therefore, slow. An efficient parallel GBIS implementation must assign and maintain an equal workload on each processor (see Fig. 3.1C).

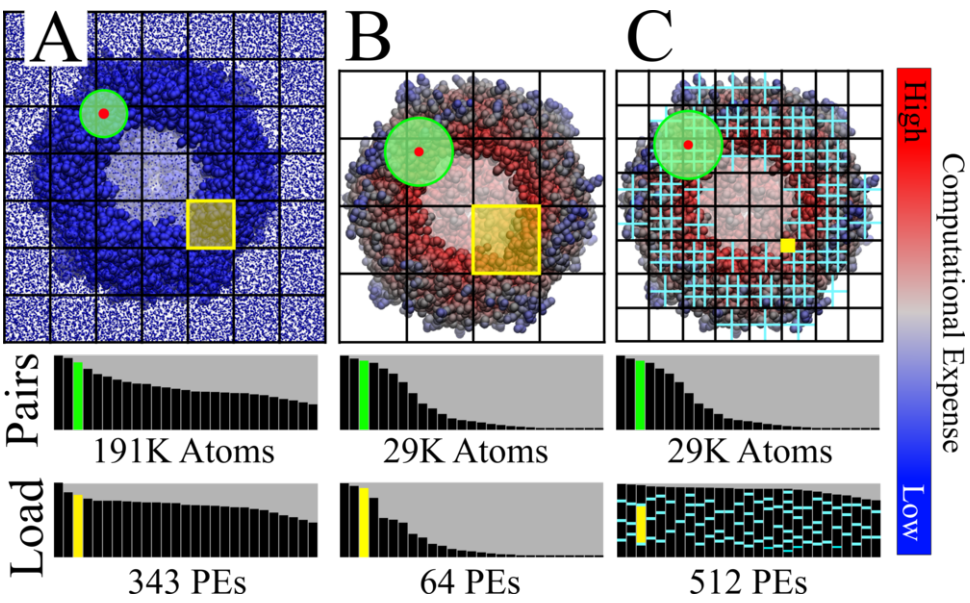

Figure 3.1: Work decomposition for implicit solvent and explicit solvent simulations. A SEp22 dodecamer is shown (front removed to show interior) with overlaid black grid illustrating domain decomposition for explicit solvent (A), implicit solvent (B) and NAMD's highly parallel implicit solvent (C). Atoms are colored according to relative work required to calculate the net force with blue being least expensive and red being most expensive. The number of neighbor-pairs within the interaction cutoff (green circle, Pairs) for an atom (red circle) varies more in implicit solvent than explicit solvent, as does the number of atoms within a spatial domain (yellow box, Load), each domain being assigned to a single processor (PE). (A) Because explicit solvent has a spatially homogeneous distribution of atoms, it has a balanced work load among processors using simple domain decomposition. (B) Domain decomposition with implicit solvent suffers from the spatially heterogenous atom distribution; the work load on each processor varies highly. (C) NAMD's implicit solvent model (cyan grid representing force decomposition and partitioning), despite having varying number of atoms per domain and varying computational cost per atom, still achieves a balanced workload among processors.

3.2.5 *Challenge 3: Three iterations over atom-pairs per timestep*

Instead of requiring one iteration over atom-pairs to calculate electrostatic forces, GBIS requires three independent iterations over atom-pairs (c.f. eqs. 3.9, 3.11 and 3.12). Because each of these iterations depends on the previous iteration, the cost associated with communication and synchronization is tripled. An efficient parallel GBIS implementation must schedule communication and computation on each processor as to maximize efficiency.

3.2.6 Parallelization Strategy

NAMD's unique strategy [114] for fast parallel MD simulations [41] enables it to overcome the three challenges of parallel GBIS simulations. NAMD divides GBIS calculations into many small work units using a three-tier decomposition scheme, assigns a balanced load of work units to processors, and schedules work units on each processor to maximize efficiency.

NAMD's three-tier work decomposition scheme [115] directly addresses Challenge 1 of parallel GBIS calculations. NAMD first employs domain decomposition to initially divide the system into a three-dimensional grid of spatial domains. Second, NAMD assigns a force work unit to calculate pairwise forces within each domain and between each pair of adjacent domains. Third, each force work unit is further partitioned into up to ten separate work units, where each partition calculates only one tenth of the atom-pairs associated with the force work unit. Dividing force work units into partitions based on computational expense avoids the unnecessary communication overhead arising from further partitioning already inexpensive force work units belonging to under populated domains. Adaptively partitioning the force work units based on computational expense improves NAMD's parallel performance even for non-implicit solvent simulations. NAMD's decomposition scheme is able to finely divide simulations into many ($\sim$40,000 for SEp22 dodecamer) small work units and, thereby, utilize thousands of processors.

NAMD's load balancer, a tool employed to ensure each processor carries an equivalent workload, initially distributes work units evenly across processors, thus partially overcoming Challenge 2 of parallel GBIS calculations. However, as atoms move during a simulation, the number of atoms in each domain fluctuates (more so than for the explicit solvent case) which causes the computational workload on each processor to change. NAMD employs a measurement based load balancer to maintain a uniform workload across processors during a simulation; periodically, NAMD measures the computational cost associated with each work unit and redistributes work units to new processors as required to maintain a balanced workload among processors. By continually balancing the workload, NAMD is capable of highly efficient simulations despite spatially heterogeneous atom distributions common to implicit solvent descriptions.

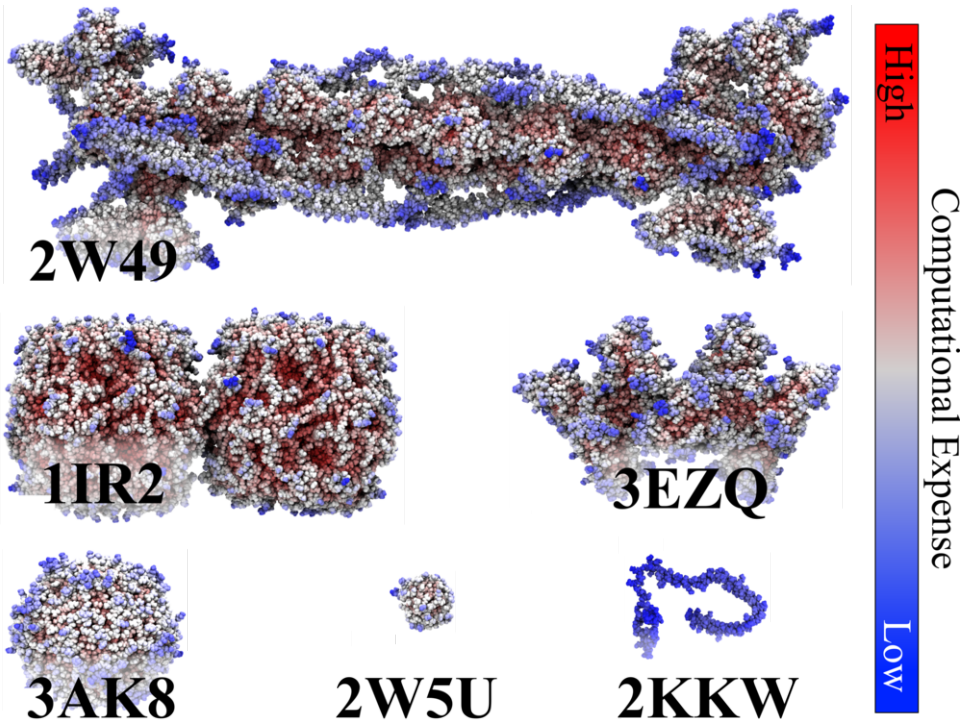

Figure 3.2: Biomolecular systems in benchmark. The performance of NAMD's parallel GBIS implementation was tested on six structures (Protein Data Bank IDs shown); the number of atoms and interactions are listed in Table 3.2. To illustrate the spatially heterogeneous distribution of work, each atom is colored by the relative time required to compute its net force with blue being fastest and red being slowest.

Though the three iterations over atom-pairs hurt parallel efficiency by requiring additional (compared to the explicit solvent case) communication and synchronization during each timestep, NAMD's unique communication scheme is able to maintain parallel efficiency. Unlike most MD programs, NAMD is capable of scheduling work units on each processor in an order which overlaps communication and computation, thus maximizing efficiency. NAMD's overall parallel strategy of work decomposition, workload balancing and work unit scheduling permits fast and efficient parallel GBIS simulations of even very large systems.

3.2.7 Performance Benchmark

To demonstrate the success of NAMD's parallel GBIS strategy, protein systems of varying sizes and configurations were simulated on 2-2048 processor cores using NAMD version 2.8. We also compare against an implementation of domain decomposition taken from Amber's PMEMD version 9 [40], which also contains the original implementation of the GB^{OBC} implicit solvent model [53]. The benchmark consists of six systems, listed in Table 3.2 and displayed in Fig. 3.2, chosen to represent small (2,000 atoms), medium (30,000 atoms) and large (150,000 atoms) systems.

The following simulation parameters were employed for the benchmark simulations. A value of 16 Å was used for both nonbonded interaction cutoff and the Born radius calculation cutoff. An implicit ion concentration of 0.3 M was assumed. A time step of 1 fs was employed with all forces being evaluated every step. System coordinates were not written to a trajectory file. For the explicit solvent simulation (Table 3.2: 3AK8-E), nonbonded interactions were cutoff and smoothed between 10 and 12 Å, with PME [116] electrostatics, which require periodic boundary conditions, being evaluated every four steps.

Simulations were run on 2.3 GHz processors with 1 GB/s network interconnect for 600 steps. NAMD's speed is reported during simulation and was averaged over the last 100 steps (first 500 steps are dedicated to initial load balancing). The speed of the domain decomposition implementation, in units seconds per timestep, was calculated as ("Master NonSetup CPU time")/(total steps); simulating up to 10,000 steps did not return noticeably faster speeds. Table 3.2 reports the simulation speeds in seconds/step for each benchmark simulation; Fig. 3.3 presents simulation speeds scaled by system size in terms of the number of pairwise interactions per second (pips) calculated.

	NAMD	Amber	error
2KKW	-5,271.42	-5,271.23	3.6E-05
2W5U	-6,246.10	-6,245.88	3.5E-05
3EZQ	-94,291.43	-94,288.17	3.4E-05
3AK8	-89,322.23	-89,319.13	3.4E-05
2W49	-396,955.31	-396,941.54	3.4E-05
1IR2	-426,270.23	-426,255.45	3.4E-05

Table 3.1: Total electrostatic energy of benchmark systems. To validate NAMD's implicit solvent implementation, the total electrostatic energy (in units kcal/mol) of the six benchmark systems was calculated by NAMD and Amber implementations of the GB^{OBC} implicit solvent [53]; error is calculated through eq. 3.13.

3.2.8 Implementation Validation

The correctness of our GBIS implementation was validated by comparison to the method's [53] original implementation in Amber [40]. Comparing the total electrostatic energy (see eq. 3.1) of the six test systems as calculated by NAMD and Amber demonstrates their close agreement. Indeed, Table 3.1 shows that the relative error, defined through

$$\text{error} = \left|(E_{\text{T}}^{\text{Elec}}(\text{NAMD}) - E_{\text{T}}^{\text{Elec}}(\text{Amber}))/E_{\text{T}}^{\text{Elec}}(\text{Amber})\right| \ , \tag{3.13}$$

is less than 4×10^{-5} for all structures in Fig. 3.2.

3.2.9 Molecular Dynamics Flexible Fitting of Ribosome

To illustrate the utility of NAMD's parallel GBIS implementation, we simulate the *Escherichia coli* ribosome. The ribosome is the cellular machine that translates genetic information on mRNA into protein chains.

During translation, tRNAs, with their anti-codon loops to be matched to the genetic code on mRNA, carry amino acids to the ribosome. The synthesized protein chain is elongated by one amino acid each time a cognate tRNA (with its anti-codon loop complementary to the next mRNA codon) brings an amino acid to the ribosome; a peptide bond is formed between the new amino acid and the existing protein chain. During protein synthesis, the ribosome complex fluctuates between two conformational states, namely the so-called classical and ratcheted state [44].

During transition from classical to ratcheted state, the ribosome undergoes multiple, large conformational changes, including an inter-subunit rotation between its 50S and 30S subunits [44] and the closing of its L1 stalk in the 50S subunit [45] (see Fig. 3.4). The large conformational changes during the transition from classical to ratcheted state are essential for translation [46] as suggested by previous cryo-EM data [47]. To demonstrate the benefits of NAMD GBIS, we simulate the large conformational changes during ratcheting of the ∼250,000-atom ribosome using molecular dynamics flexible fitting.

The molecular dynamics flexible fitting (MDFF) method [3, 117, 118] is a MD simulation method that matches crystallographic structures to an electron microscopy (EM) map; crystallographic structures often correspond to non-physiological states of biopolymers while EM maps correspond often to functional intermediates of biopolymers. MDFF-derived models of the classical and ratcheted state ribosome provide atomic-level details crucial to understanding protein elongation in the ribosome. The MDFF method adds to a conventional MD simulation an EM map-derived potential, thereby driving a crystallographic structure towards the conformational state represented by an EM map. MDFF was previously applied to successfully match crystallographic structures of the ribosome to ribosome functional states as seen in EM [119–123]. Shortcomings of MDFF are largely due to the use of in vacuo simulations; such use was necessary hitherto as simulations in explicit solvent proved too cumbersome. Implicit solvent MDFF simulations promise a significant improvement of the MDFF method. We applied MDFF here, therefore, to fit an atomic model of a classical state ribosome into an EM map of a ratcheted state ribosome [47].

The classical state in our simulations is an all-atom ribosome structure [124] with 50S and 30S subunits taken from PDB IDs 2I2V and 2I2U, respectively [125], and the complex fitted to an 8.9 Å resolution classical state EM map [47]. In the multistep protocol for fitting this classical state ribosome to a ratcheted state map [117], the actual ribosome is fitted first, followed by fitting the tRNAs. Since the fitting of the ribosome itself exhibits the largest conformational changes (inter-subunit rotation and L1-stalk closing), we limit our

MDFF calculation here to the ribosome and do not include tRNAs.

Three MDFF simulations were performed using NAMD [41] and analyzed using VMD [81]. The MDFF simulations are carried out in explicit TIP3P [83] solvent, in implicit solvent and in vacuo. All simulations were performed in the NVT ensemble with the AMBER99 force field [126], employing the SB [127] and BSC0 [128] corrections and accounting for modified ribonucleosides [129]. The grid scaling parameter [3], which controls the balance between MD force field and the EM-map derived force field, was set to 0.3. Simulations were performed using a 1 fs timestep with nonbonded forces being evaluated every two steps. Born radii were calculated using a cutoff of 14 Å, while the nonbonded forces were smoothed and cut off between 15 and 16 Å. An implicit ion concentration of 0.1 M was assumed with protein and solvent dielectric set to 1 and 80, respectively. A Langevin thermostat with a damping coefficient of 5 ps^{-1} was employed to hold the temperature to 300 K. In the explicit solvent simulation, the ribosome was simulated in a periodic box of TIP3P water [83] including an explicit ion concentration of 0.1 M, with nonbonded forces cut off at 10 Å and long-range electrostatics calculated by PME every four steps. The in vacuo simulation utilized the same parameters as explicit solvent, but without inclusion of solvent or bulk ions, and neither PME nor periodicity were employed.

Each system was minimized for 5000 steps before performing MDFF for 3 ns. For the explicit solvent simulation, an additional 0.5 ns equilibration of water and ions was performed, with protein and nucleic acids restrained, before applying MDFF.

To compare behavior of solvent models during the ribosome simulations, we calculate the root-mean-square deviation between models as

$$\mathrm{RMSD}_{\mathrm{sol,ref}}(t) = \sqrt{\sum_{i}^{N} \left[\vec{r}_{i,\mathrm{sol}}(t) - \vec{r}_{i,\mathrm{ref}}\right]^2 / N} \,, \tag{3.14}$$

where $\vec{r}_{i,\mathrm{sol}}(t)$ denotes the atomic coordinates at time t of the simulation corresponding to one of the three solvent models (exp, imp or vac) and $\vec{r}_{i,\mathrm{ref}}$ denotes the atomic coordinates for the last time step ($t_f = 3$ ns) of the simulation using the reference solvent model (exp, imp or vac) as specified below. Unless otherwise specified, the summation is over the $N = 146,000$ heavy atoms excluding the mRNA, L10 and L12 protein segments which are too flexible to be resolved by the cryo-EM method.

3.3 Results

3.3.1 Performance Benchmarks

The results of the GBIS benchmark simulations are listed in Table 3.2. Fig. 3.3 illustrates the simulation speeds, scaled by system size, as the number of pairwise interactions per second (pips) calculated. For a perfectly efficient algorithm, pips would be independent of system size or configuration and would increase proportionally with processor count. NAMD's excellent parallel GBIS performance is demonstrated as pips is nearly the same for all six systems and increases almost linearly with processor count as highlighted (in blue) in Fig. 3.3.

The domain decomposition algorithm [40] performs equally well for small systems, but its performance suffers significantly when system size and processor count are increased. The domain decomposition implementation also appears to be limited to a pips maximum of 10^8 pairs/sec across all system sizes, no matter how many processors are used, as highlighted (in red) in Fig. 3.3. NAMD runs efficiently on twice the number of processors compared to domain decomposition and greatly outperforms it for the large systems tested. The SEp22 dodecamer timings for both implicit (3AK8) and explicit (3AK8-E) solvent reported in Table 3.2 demonstrate that NAMD's parallel GBIS implementation is as efficient as its parallel explicit solvent capability. We note that the simulation speed for GBIS can be further increased, without significant loss of accuracy, by shortening either the interaction or Born radius calculation cutoff distance.

3.3.2 Ribosome Simulation

To demonstrate the benefit of NAMD's GBIS capability for simulating large structures, a high-resolution classical state ribosome structure was fitted into a low-resolution ratcheted state EM map in an in vacuo MDFF simulation as well as MDFF simulations employing explicit and implicit solvent. During the MDFF simulation, the ribosome undergoes two major conformational changes: closing of the L1 stalk and rotation of the 30S subunit relative to the 50S subunit (see Fig. 3.4).

To compare the rate of convergence and relative accuracy of solvent models, the RMSD values characterizing the three MDFF simulations were calculated using eq. Fig. 3.14. Fig. 3.4 plots $\mathrm{RMSD}_{\mathrm{exp,exp}}(t)$, $\mathrm{RMSD}_{\mathrm{imp,exp}}(t)$ and $\mathrm{RMSD}_{\mathrm{vac,exp}}(t)$ that compare each MDFF simulation against the final structure reached in the explicit solvent case. We note that using the initial rather than final structure as the reference could yield a slightly different characterization of convergence [130], e.g., a slightly different convergence time. As manifested by $\mathrm{RMSD}_{\mathrm{imp,exp}}(t)$ and $\mathrm{RMSD}_{\mathrm{vac,exp}}(t)$, the implicit solvent and vacuum MDFF calculations converge to their respective final structures in 0.5 ns compared to $\sim$1.5-2 ns for the explicit solvent case, i.e.,

PDB ID	2KKW	2W5U	3EZQ	3AK8	3AK8-E	2W49	1IR2
atoms	2,016	2,412	27,600	29,479	191,686	138,136	149,860
pairs	0.25 M	1 M	13.2 M	15.7 M	63.8 M	65.5 M	99.5 M

# procs	NAMD sec/step						
2	0.0440	0.167	2.01	2.87	2.25	10.4	16.3
4	0.0225	0.0853	1.00	1.44	1.12	5.47	8.70
8	0.0122	0.0456	0.505	0.726	0.568	2.61	4.09
16	0.00664	0.0228	0.260	0.371	0.286	1.31	2.05
32	0.00412*	0.0126	0.136	0.191	0.146	0.661	1.03
64	-	0.00736*	0.0700	0.105	0.0868	0.333	0.520
128	-	-	0.0447	0.0575	0.0523	0.169	0.267
256	-	-	0.0321	0.0340	0.0288*	0.0935	0.148
512	-	-	0.0224*	0.0171*	-	0.0618	0.0806
1024	-	-	-	-	-	0.0461*	0.0563*
2048	-	-	-	-	-	0.0326	0.0486

# procs	Domain Decomposition sec/step						
2	0.0613	0.208	5.57	7.15	-	306	373
4	0.0312	0.104	2.83	3.67	-	169	203
8	0.0163	0.0535	1.43	1.84	-	92.5	111
16	0.0090*	0.0277	0.727	0.934	-	51.6	62.0
32	0.0070	0.0162*	0.391	0.506	-	25.9	31.3
64	-	0.013	0.254*	0.307*	-	13.1	15.7
128	-	-	0.191	0.220	-	6.74	8.08
256	-	-	-	-	-	3.63	4.28
512	-	-	-	-	-	2.16*	2.66*
1024	-	-	-	-	-	1.95	2.07

Table 3.2: NAMD and domain decomposition benchmark data. Speed, in units seconds/step, for both NAMD and the domain decomposition algorithm for the six test systems (see Fig. 3.2) on 2-2048 processors (procs). Also listed are the number of atoms and pairs of atoms (in millions, M) within the cutoff (16 Å for implicit solvent and 12 Å for explicit solvent) in the initial structure. Data is not presented for higher processor counts with slower simulation speeds. An asterisk marks highest processor count for which doubling the number of processors increased simulation speed by at least 50%. 3AK8-E uses explicit solvent. Each simulated system demonstrates that NAMD can efficiently utilize at least twice the number of processors as domain decomposition and, thereby, achieves simulation speeds much faster than for domain decomposition.

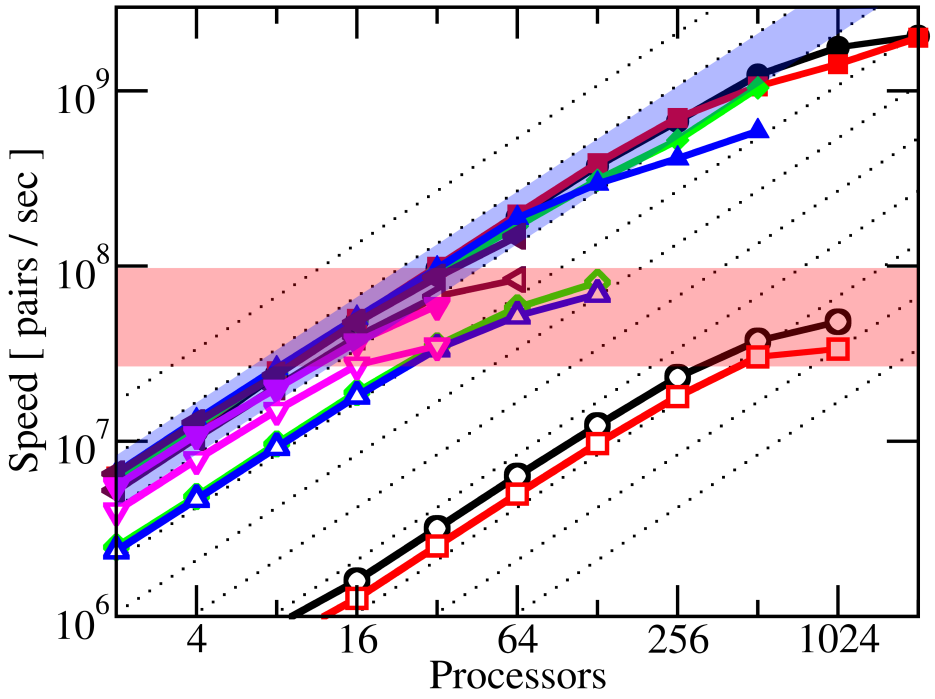

Figure 3.3: Parallel performance of NAMD and domain decomposition implicit solvent. Computational speed (pairwise interactions per second) for six biomolecular systems (see Fig. 3.2): 1IR2 (black circle), 2W49 (red square), 3AK8 (green diamond), 3EZQ (blue up triangle), 2W5U (maroon left triangle), and 2KKW (magenta down triangle). NAMD's parallel implicit solvent implementation (solid shapes) performs extremely well in parallel as seen by performance increasing linearly (blue highlight) with number of processors and is independent of system size. Performance of domain decomposition implicit solvent, however, suffers in parallel (empty shapes). Not only does there appear to be a maximum speed of 10^8 pairs/sec (red highlight) regardless of processor count, but the large systems (1IR2, 2W49) also perform at significantly lower efficiency than the small systems (2E5U, 2KKW). Diagonal dotted lines represent perfect speedup. See also Table 3.2.

for $\mathrm{RMSD}_{\mathrm{exp,exp}}(t)$.

The final structures obtained from the MDFF simulations are compared in Table 3.3 through the $\mathrm{RMSD}_{\mathrm{sol,ref}}(t)$ values for $t = 3$ ns. The ribosome structure from GBIS MDFF closely agrees with the one from explicit solvent MDFF as indicated by the value $\mathrm{RMSD}_{\mathrm{imp,exp}}(3\,\mathrm{ns}) = 1.5$ Å; the in vacuo MDFF ribosome structure, however, compares less favorably with the explicit solvent MDFF structure as suggested by the larger value $\mathrm{RMSD}_{\mathrm{vac,exp}}(3\,\mathrm{ns}) = 1.9$ Å. While the 0.4 Å improvement in RMSD of the GBIS MDFF, over in vacuo MDFF, structure implies an overall enhanced quality, certain regions of the ribosome are particularly improved.

The regions with the highest structural improvement (highlighted red in Fig. 3.4) belong to segments at the exterior of the ribosome and to segments not resolved by and, therefore, not coupled to the EM map, i.e., not being directly shaped by MDFF. For proteins at the exterior of the ribosome, GBIS MDFF produces higher quality structures than in vacuo MDFF, because these proteins are highly exposed to solvent and, therefore, require a solvent description. The structural improvement for several exterior solvated proteins, calculated by $\mathrm{RMSD}_{\mathrm{vac,exp}}(3\,\mathrm{ns}) - \mathrm{RMSD}_{\mathrm{imp,exp}}(3\,\mathrm{ns})$, is 3.5 Å , 2.4 Å and 1.6 Å for ribosomal proteins S6, L27 and S13 (highlighted red in Fig. 3.4), respectively. Accurate modeling of these proteins is critical for studying the translation process of the ribosome. The L27 protein, for example, not only facilitates the assembly of the 50S subunit, it also ensures proper positioning of the new amino acid for peptide bond formation [131]. The S13 protein, located at the interface between subunits, is critical to the control of mRNA and tRNA translocation within the ribosome [132].

The use of GBIS for MDFF also increases structural quality in regions where the EM map does not resolve the ribosome's structure and, therefore, MDFF does not directly influence conformation; though it is most important that MDFF correctly models structural regions defined in the EM map, it is also desirable that it correctly describes regions of crystal structures not resolved by the EM map. The structural improvement, over in vacuo MDFF, of the unresolved segments is 8.3 Å for mRNA and 4.9 Å for L12 (highlighted red in Fig. 3.4). The L12 segment is a highly mobile ribosomal protein in the 50S subunit that promotes binding of factors which stabilize the ratcheted conformation; L12 also promotes GTP hydrolysis which leads to mRNA translocation [133]. As clearly demonstrated, the use of GBIS MDFF, instead of in vacuo MDFF, improves the MDFF method's accuracy for matching crystallographic structures to EM maps, particularly for highly solvated or unresolved proteins.

To compare computational performance of the solvent models for MDFF, each ribosome simulation was benchmarked on 1020 processor cores (3.5 GHz processors with 5 GB/s network interconnect); the simulation speed for explicit solvent MDFF is 3.6 ns/day, for implicit solvent MDFF it is 5.2 ns/day and for vacuum

MDFF it is 37 ns/day. GBIS MDFF performs 50% faster than does explicit solvent MDFF, but seven times slower than in vacuo MDFF. NAMD's GBIS implementation is clearly able to achieve a more accurate MDFF match of the ribosome structure (see Table 3.3) than does an in vacuo MDFF calculation and does so at a lower computational cost than explicit solvent MDFF.

		Reference		
		exp	imp	vac
Solvent	exp	0	1.5	1.9
	imp	1.5	0	2.1
	vac	1.9	2.1	0

Table 3.3: Root-mean-square deviation ($\mathrm{RMSD}_{\mathrm{sol,ref}}(3\,\mathrm{ns})$ in Å) between the three final ribosome structures matched using explicit solvent, GBIS, and in vacuo MDFF. The GBIS and explicit solvent MDFF structures closely agree as seen by $\mathrm{RMSD}_{\mathrm{imp,exp}}(3\,\mathrm{ns}) = 1.5$ Å, while the in vacuo MDFF structure deviates from the explicit solvent MDFF structure by $\mathrm{RMSD}_{\mathrm{vac,exp}}(3\,\mathrm{ns}) = 1.9$ Å. See also Fig. 3.4.

3.4 Summary

The generalized Born implicit solvent (GBIS) model has long been employed for molecular dynamics simulations of relatively small bio-molecules. NAMD's unique GBIS implementation can also simulate very large systems, such as the entire ribosome, and does so efficiently on large parallel computers. The new GBIS capability of NAMD will be beneficial to accelerate in simulations the slow motions common to large systems by eliminating viscous drag from water.

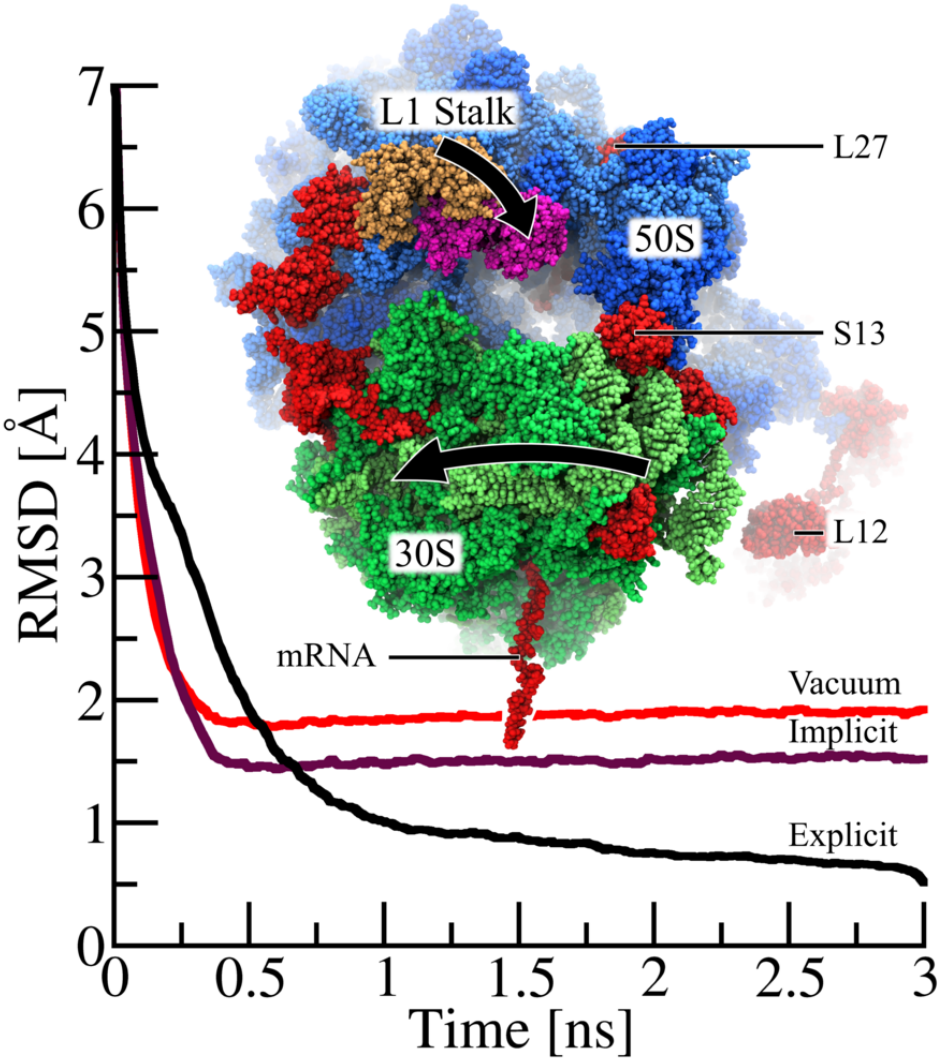

Figure 3.4: Molecular dynamics flexible fitting (MDFF) of ribosome with NAMD's GBIS method. While matching the 250,000-atom classical ribosome structure into the EM map of a ratcheted ribosome, the 30S subunit (green) rotates relative to the 50S subunit (blue) and the L1 stalk moves 30 Å from its classical (tan) to its ratcheted (magenta) position. Highlighted (red) are regions where the implicit solvent structure agrees with the explicit solvent structure much more closely than does the in vacuo structure. The root-mean-squared deviation ($\mathrm{RMSD}_{\mathrm{sol,exp}}(t)$) of the ribosome, defined in eq. 3.14, with the final fitted explicit solvent structure as reference, is plotted over time for explicit solvent ($\mathrm{RMSD}_{\mathrm{exp,exp}}(t)$ in black), implicit solvent ($\mathrm{RMSD}_{\mathrm{imp,exp}}(t)$ in purple) and in vacuo ($\mathrm{RMSD}_{\mathrm{vac,exp}}(t)$ in red) MDFF. While the explicit solvent MDFF calculation requires ∼1.5-2 ns to converge to its final structure, both implicit solvent and vacuum MDFF calculation require only 0.5 ns to converge. As seen by the lower RMSD values for $t > 0.5$ ns, the structure derived from the implicit solvent fitting agrees more closely with the final explicit solvent structure than does the in vacuo structure. While this plot illustrates only the overall improvement of the implicit solvent structure over the in vacuo structure, the text discusses key examples of ribosomal proteins (L27, S13 and L12) whose structural quality is significantly improved by the use of implicit solvent.

Chapter 4

Hybrid GPU/CPU Algorithm for GB/SA Implicit Solvent Calculations

4.1 Introduction

Laboratory and clinical studies of living cells are increasingly complemented by computational atomic level modeling of cellular processes [6, 7, 134–139]. In fact, modeling has developed today into a computational microscope [1] used to explore nanoscale structures and processes in structural cell biology [117, 140, 141], cellular mechanics [1, 8, 142], and nanosensor development [143, 144]. Small size and short time scales of many sub-cellular processes challenge experimental methodologies, making the molecular dynamics computational microscope an ideal supporting tool for biological research.

Molecular dynamics (MD) has already enabled, in particular, modeling of health-relevant biomolecular systems and processes, e.g., viral infection [4, 5], interactions between tissues and therapeutics [6, 7], blood coagulation [8–15], and amyloid fibril formation [16–21]. Often, however, simulation time scales are too short to be of value; only by continually adopting the latest computing technologies can simulations speeds increase, and extend the reach of MD to the millisecond-scale [22] that is necessary to describe many cellular processes.

MD simulations calculate interatomic forces of biopolymer systems and solve the classical equations of motion for each time step [41], to explore the dynamic behavior of biological systems. The largest computational expense in MD simulations stems from calculating so called "non-bonded" interactions, forces between atoms which are not covalently bonded, comprising Coulomb and van der Waals forces.

The calculation of non-bonded forces is highly amenable to parallel computing [41, 115] because the modeled interaction between two atoms is independent of all other atoms, thus, atom-pair interactions can be calculated independently from one another, then accumulated to determine net atomic forces. These independent calculations offer a high degree of concurrency, making them ideal for parallel computing; recently, NAMD successfully conducted a 100,000,000-atom MD simulation on 200,000 CPU cores [145]. Along with multi-core CPUs, MD programs have also harnessed new computing technologies such as graphics processing units (GPUs) [146, 147].

4.1.1 Graphics Processing Units

Though originally developed only for high-quality graphics rendering, GPUs are well suited for accelerating general scientific calculations. A single GPU may today contain 16 multiprocessors, each with 32 cores, yielding 512 cores per GPU. Maximizing the use of such highly parallel processors requires meeting strict criteria such as coalescing access to slow memory and issuing 10,000s of independent calculations. Because many molecular biological calculations can meet the strict criteria, GPUs have been used to accelerate biological computing applications such as molecular modeling [48], electrostatic calculations [49, 146], molecular orbital display [50] and simulations of protein diffusion in whole cells [51].

The application of GPUs to molecular dynamics has already demonstrated significant performance benefits [148] for both explicit solvent [147, 149] and implicit solvent [150, 151] models. Each implementation varies in the use of the GPU, ranging from employing the GPUs for calculating forces only [147] to using them to integrate the equations of motion [152, 153] as well. GPU implementations also vary in regard to how restrictive they are of the MD methodologies they admit; for example, some implementations allow a non-bonded interaction cut-off length [146] which is common for large systems, while others do not [150], thereby being applicable only to small systems.

The great success already achieved in accelerating full molecular dynamics calculations with GPUs inspires additional work in advancing MD calculations which involve both GPU-accelerated calculations and CPU calculations (those not yet adapted to the GPU) as they arise in, for example, steering [154] or grid potentials [118]. Prior efforts at GPU-acceleration of MD simulations have adapted almost all of the computation to the GPU, the CPU's role remaining secondary [152, 153].

4.1.2 Hybrid GPU/CPU Computing

Each GPU requires a so-called host CPU to send data to and receive data from the GPU as well as instruct the GPU as to which calculations to perform. Therefore, a necessary and difficult component of hybrid GPU/CPU calculations is allowing host CPUs to switch between GPU-hosting responsibilities and performing their own scientific calculations. Because many existing MD methods have not yet been adapted to GPUs, or are not amenable to GPU calculation, it is vital to explore how to efficiently perform hybrid calculations which require full use of both GPUs and CPUs and which, therefore, require coordination between the two.

An ideal application for hybrid computing is the generalized Born (GB) implicit solvent with solvent-accessible surface area (SASA or SA) calculation, commonly abbreviated GB/SA. A GB/SA implicit solvent simulation employs the GB calculation to account for the polar, i.e., hydrophilic, effects of water while

the SA calculation models the nonpolar, i.e., hydrophobic, effects. It is known that the hydrophobic free energy of solvation is approximately equal to the product of the molecular surface area and a surface tension parameter [52, 54]. The GB/SA calculation consists of three classes of non-bonded interatomic forces: the classical Coulomb and van der Waals forces, the generalized Born [52, 53, 113] (GB) force, and the solvent-accessible surface area (SA) force, calculated via the linear combination of pairwise overlaps [54] (LCPO) algorithm, as in the Amber MD program [155]. While the Coulomb, van der Waals and GB force calculations are being calculated on the GPU, the LCPO algorithm's SA calculation is best performed on the CPU, thus making the GB/SA calculation an ideal candidate for hybrid GPU/CPU calculation.

The present work explores a variety of performance and functionality issues relevant to GPU accelerated calculations of the GB/SA implicit solvent model in the context of the NAMD parallel MD program [41]. First, we analyze the necessary non-bonded force calculations and describe why each is best suited for GPUs or CPUs. Second, we outline NAMD's algorithmic strategy for GB/SA calculations on hybrid GPU/CPU computers. Finally, we demonstrate the performance of the hybrid strategy through ∼250 benchmark simulations and explore the biological importance of including SA calculations.

4.2 Methods

The generalized Born / solvent-accessible surface area (GB/SA) implicit solvent model constitutes a fast representation of a solvent's polar and nonpolar effects on biomolecules [52]. Fast simulation speed can be attained by calculating the Coulomb, van der Waals and generalized Born (GB) forces on the GPU while simultaneously calculating the hydrophobic surface area (SA) force on the CPU. Here, we present the equations arising in the Coulomb, van der Waals, GB and SA force calculations which motivate the hybrid GPU/CPU algorithm employed. Next, NAMD's implementation of the various calculations will be described.

4.2.1 Coulomb and Van der Waals Forces

The non-bonded forces arising in explicit and implicit solvent simulations are the Coulomb and van der Waals (calculated using the Lennard-Jones description) forces. The total system energy contributed by Coulomb and van der Waals interactions, $E_{\mathrm{T}}^{\mathrm{CW}}$, is

$$E_{\mathrm{T}}^{\mathrm{CW}} \;=\; (1/2)\sum_{i}\sum_{j\in N(i)}\{\underbrace{4\epsilon_{ij}\left[(\sigma_{ij}/r_{ij})^{12}-(\sigma_{ij}/r_{ij})^{6}\right]}_{\text{van der Waals}}+\underbrace{(k_{\mathrm{e}}/\epsilon_{\mathrm{p}})\,(q_iq_j/r_{ij})}_{\text{Coulomb}}\}\,, \tag{4.1}$$

where ϵ_{ij} and σ_{ij} are the well depth and equilibrium interaction length parameter of the Lennard-Jones potential, r_{ij} is the distance between atoms i and j, $k_{\mathrm{e}} = 332$ kcal Å/e^2 the Coulomb constant, $\epsilon_{\mathrm{p}} = 1$ the dielectric constant of the protein interior and q_i the atomic charge; $N(i)$ is the set of all neighbors, j, that are within the interaction cut-off, r_c, from atom i.

The net force, $\vec{F}_i$, on an atom is calculated as

$$\vec{F}_i \;=\; -\sum_{j \in N(i)} (dE_{\mathrm{T}}/dr_{ij})\hat{r}_{ij} \;; \tag{4.2}$$

by applying Eq. 4.2 to Eq. 4.1, the net Coulomb and van der Waals forces on an atom, $\vec{F}_i^{\mathrm{CW}}$, is

$$\vec{F}_i^{\mathrm{CW}} \;=\; \sum_{j \in N(i)} \{\underbrace{24\epsilon_{ij}\left[2\left(\sigma_{ij}^{12}/r_{ij}^{13}\right) - \left(\sigma_{ij}^{6}/r_{ij}^{7}\right)\right]}_{\text{van der Waals}} + \underbrace{\left(k_{\mathrm{e}}/\epsilon_{\mathrm{p}}\right)\left(q_i q_j/r_{ij}^2\right)}_{\text{Coulomb}}\}\hat{r}_{ij} \;. \tag{4.3}$$

The summation in Eq. 4.3 requires an MD program to iterate over all pairs of interacting atoms, i and j, where $r_{ij} < r_c$; the successful application of GPUs to computing atom-pair interactions, such as Coulomb and van der Waals forces, has previously been demonstrated in NAMD [146, 147].

4.2.2 Generalized Born Implicit Solvent Forces

The generalized Born implicit solvent model [52, 53], already implemented in NAMD for the CPU [43], describes water as a bulk solvent acting as a dielectric [52, 53]; the dielectric solvent screens, i.e., reduces, electrostatic interactions between charged atoms. The total GB energy, i.e., hydrophilic energy of solvation, of the atomic system is given by the sum of pair- and self-energies according to

$$E_{\mathrm{T}}^{\mathrm{GB}} = (1/2)\sum_i \sum_{j \in N(i)} \underbrace{E_{ij}^{\mathrm{GB}}}_{\text{pair}} + (1/2)\sum_i \underbrace{E_{ii}^{\mathrm{GB}}}_{\text{self}} \,, \tag{4.4}$$

where the pair- and self-GB energies are calculated according to

$$E_{ij}^{\mathrm{GB}} = -k_e D_{ij} q_i q_j / f_{ij}^{\mathrm{GB}} \;. \tag{4.5}$$

Here, D_{ij} is an effective dielectric constant [108] between atoms i and j, and f_{ij}^{GB} is [52]

$$f_{ij}^{\mathrm{GB}} = \sqrt{r_{ij}^2 + \alpha_i \alpha_j \exp\left(-r_{ij}^2/4\alpha_i\alpha_j\right)} \;. \tag{4.6}$$

The quantities α_i arising in this expression, the so-called atomic Born radii, describe an atom's exposure to solvent and, thus, characterize the degree to which an atom's electrostatic interaction is screened; α_i is calculated, according to Onufriev, Bashford and Case's (OBC) description [53], as

$$\alpha_i = \left[1/(\rho_i - 0.09\ \text{Å}) - (1/\rho_i)\tanh\left(\delta\psi_i - \beta\psi_i^2 + \gamma\psi_i^3\right)\right]^{-1}\ , \tag{4.7}$$

where ρ_i is the atomic radius as parameterized by the OBC model; $\delta = 1$, $\beta = 0.8$ and $\gamma = 4.85$ are additional dimensionless parameters of the OBC model [53] which enable calculation of the correct Born radii, i.e., those derived from Poisson-Boltzmann calculations, from the sum, ψ_i, of pairwise atomic descreening, H_{ij},

$$\psi_i = (\rho_i - 0.09\ \text{Å}) \sum_{j \in N(i)} H_{ij}\ . \tag{4.8}$$

The pairwise descreening, H_{ij}, between neighboring atoms i and j is given by a distance-dependent piecewise function, defined in four mutually exclusive interaction domains,

$$H_{ij} = \begin{cases} 0 & (r_{ij} > r_c + \rho_j) \\ f_1(r_{ij}, \rho_j, r_c) & (r_{ij} > r_c - \rho_j) \\ f_2(r_{ij}, \rho_j) & (r_{ij} > 4\rho_j) \\ f_3(r_{ij}, \rho_j) & (r_{ij} > \rho_i - 0.09\ \text{Å} + \rho_j) \\ f_4(r_{ij}, \rho_i, \rho_j) & \text{otherwise} \end{cases} \tag{4.9}$$

where f_{1-4} are also taken from the OBC model [53].

Because the GB energy defined in Eq. 4.4 depends on the interatomic distances directly, through f_{ij}^{GB}, and also indirectly, through α_i, calculating the GB force on an atom, $\vec{F}_i^{GB}$, requires multiple partial derivatives [43], namely,

$$\begin{aligned} \vec{F}_i^{GB} = \sum_{j \in N(i)} & [(\partial E_{ij}^{GB}/\partial f_{ij}^{\text{GB}})(df_{ij}^{\text{GB}}/dr_{ij}) \\ & + (\partial E_T^{GB}/\partial \alpha_i)(d\alpha_i/dr_{ij}) \\ & + (\partial E_T^{GB}/\partial \alpha_j)(d\alpha_j/dr_{ij})]\ \hat{r}_{ij}\ . \end{aligned} \tag{4.10}$$

Of the various required derivatives, $\partial E_{ij}^{GB}/\partial f_{ij}^{\text{GB}}$, $\partial E_T^{GB}/\partial \alpha_k$ and $d\alpha_k/dr_{ij}$, the most expensive to calculate

is $\partial E_T^{GB}/\partial\alpha_k$, as it requires an additional summation over atom-pairs,

$$\partial E_T^{GB}/\partial\alpha_k = (1/2)\sum_i \sum_{j\in N(i)} \left[\partial E_{ik}^{GB}/\partial\alpha_k + \partial E_{kj}^{GB}/\partial\alpha_k\right] + \sum_i \partial E_{ii}^{GB}/\partial\alpha_k \ . \tag{4.11}$$

The three summations in Eqs. 4.8, 4.10 and 4.11, known as the three phases of the GB force calculation, require three successive iterations over all pairs of atoms each time step. The three phases compute, in order, the atomic Born radii, α_i the partial derivatives $\partial E_T^{GB}/\partial\alpha_k$, and, finally, the GB force, $\vec{F}_i^{GB}$. As in the case of Coulomb and van der Waals forces, iterating over atom-pairs for the three GB phases is amenable to GPU-acceleration. Because calculating the generalized Born force as outlined in Eqs. 4.4-4.11 is several times more computationally expensive than calculating only the Coulomb and van der Waals forces, and because GPUs are well suited for such arithmetically expensive calculations, the GB calculation stands to benefit strongly from GPU-acceleration as previously demonstrated by other MD programs [150, 151].

4.2.3 Solvent-Accessible Surface Area Forces

The generalized Born model only describes the polar, i.e., hydrophilic, energy of solvation. It is desirable to account also for the nonpolar, i.e., hydrophobic, energy of solvation through a solvent-accessible surface area (SA) calculation, as it is known that the hydrophobic solvation energy is approximately proportional to SA [52, 156]. The linear combination of pairwise overlaps (LCPO) is an approximate method for calculating the SA [54] and spatial derivatives necessary for hydrophobic force calculation.

The LCPO method [54], which considers only non-hydrogen atoms, is founded on calculating the surface area overlap between two spheres representing atoms; for two spheres, centered on atoms i and j, with radii R_i and R_j, separated by a distance r_{ij}, close enough that their surfaces overlap, i.e., it must hold $r_{ij} < R_i + R_j$; the surface area of atom i overlapped by atom j, A_{ij}, is then [54]

$$A_{ij} \ = \ 2\pi R_i \left[(R_i - (r_{ij}/2) - (R_i^2 - R_j^2)/(2r_{ij})\right] \ , \tag{4.12}$$

where the radius, R_i, is the atomic van der Waals radius plus the 1.4 Å solvent probe radius [54]. According to the LCPO method, the surface area of atom i, SA_i, can be evaluated approximately by multiple overlap

summations,

$$\begin{aligned} \mathrm{SA}_i &= P_{1,i}\, 4\pi R_i^2 \\ &+ P_{2,i} \sum_{j \in N(i)} A_{ij} \\ &+ P_{3,i} \sum_{j \in N(i)} A_{ij} \sum_{k \in N(i) \cap N(j)} A_{jk} \\ &+ P_{4,i} \sum_{j \in N(i)} \left[A_{ij} \sum_{j \in N(i)} \sum_{k \in N(i) \cap N(j)} A_{jk} \right] , \end{aligned} \tag{4.13}$$

where $k \in N(i) \cap N(j)$ represents all atoms k which overlap atoms i and j, thus requiring the LCPO algorithm to iterate over atom triplets, sets of three atoms whose surfaces overlap (i.e., it must hold $r_{ij} < R_i + R_j$, $r_{ik} < R_i + R_k$, $r_{jk} < R_j + R_k$). The total molecular surface area is the sum of atomic surface areas, $\sum_i \mathrm{SA}_i$. The four atomic parameters $P_{1-4,i}$ were fitted by atom type such that the LCPO algorithm most closely approximates the correct surface area as calculated by numerical methods [54]. The hydrophobic / surface area force on atom l, $\vec{F}_l^{\mathrm{SA}}$, can be calculated employing the LCPO algorithm,

$$\vec{F}_l^{\mathrm{SA}} = -T_{\mathrm{S}} \sum_{i \in N(l)} (d\mathrm{SA}_i / dr_l) , \tag{4.14}$$

where T_{S}, with units kcal/mol/Å^2, is the surface tension parameter. The relatively inexpensive derivatives required for the SA force calculation within the LCPO algorithm were previously described [54].

For the above three force calculations, approximately 13% of the computational cost is due to Coulomb and van der Waals forces, 70% due to GB forces, and the remaining 17% due to SA forces. Having transferred most of the force calculation (Coulomb, van der Waals, GB) to the GPU, the CPU can perform the relatively inexpensive (17% of total) SA calculation in the same time as the GPU's expensive (83% of total) force calculations. The SA algorithm requires iterating over all atom triplets, compared to only pairs of atoms as in the case of GB. The summation is feasible due to the low computational cost involved and because the summation over triplets can be performed well by CPUs.

4.2.4 Incorporating the GB/SA Calculation into NAMD

To achieve a high performance implementation of the GB/SA model, formulated by Eqs. 4.1-4.14, we thought to maximize use of hybrid GPU/CPU computers as outlined. Aside from fast performance, an ideal implementation should interfere as little as possible with NAMD's already highly scalable internal structure.

To achieve advanced parallel scaling, NAMD decomposes a simulated system (Fig. 4.1A) into a 3D grid of

atom-groups (Fig. 4.1B). In the case of Coulomb, van der Waals and GB forces, atom-groups are defined such that atoms in any atom-group interact only with atoms of the same atom-group and with atoms of adjacent atom-groups. Calculating the atom-pair interactions between two neighboring atom-groups (Fig. 4.1D red and green) constitutes an independent force work unit. To illustrate our GB/SA implementation, we apply NAMD to the 13,340-atom glycogen phosphorylase protein, Protein Data Bank (PDB) ID 1GPB with missing atoms placed by VMD's [81] psfgen tool. The atoms of this protein system can be divided into roughly 100 atom-groups, shown in Fig. 4.1A, and 3,000 force work units which can be assigned across hundreds of CPU cores or GPUs [146, 147].

4.2.5 GB Force Calculation on GPUs

NAMD has already achieved a sixfold speedup by calculating Coulomb and van der Waals forces on the GPU using the CUDA language [146, 147]. Because of the success of the Coulomb and van der Waals calculations on the GPU and the close algorithmic similarities, NAMD's GB calculation on the GPU, outlined below, employs a similar GPU implementation as NAMD's previously reported Coulomb and van der Waals force calculation on the GPU [146, 147].

For parallel calculations, each processor core requires a list of instructions to drive computation; for both GPU and CPU processors, a "thread" is the list of instructions detailing what operations a processor core must perform; multi-core CPUs and GPUs both require multiple threads, i.e., multiple lists of instructions, to drive the many processor cores. In CUDA, a thread block is the group of threads which drives one of the GPU's many 32-core multiprocessors.

In NAMD, a force work unit consists of calculating the atomic interactions between two atom-groups, such as group-I (red) and group-J (green) of Fig. 4.1B and D. The calculation of the force work unit is assigned to a thread block, i.e., is calculated by a single 32-core multiprocessor, as depicted in Fig. 4.1D. Each thread (vertical line) in the thread block calculates the force on a single atom of group-I (red) due to its interactions (blue circle) with atoms of group-J (green). To accelerate memory access, the GPUs 32-core multiprocessors access group-J atoms 32 at a time, in order 1-3 shown in Fig. 4.1D.

The above pattern of NAMD's GPU calculation of Coulomb, van der Waals and GB forces exploits several features of GPU hardware to achieve fast performance. First, because adjacent threads operate on atoms adjacent in memory, reading atomic coordinates from and writing atomic forces to slow memory, is coalesced, giving significant memory speedup over non-coalesced memory access. Second, because GPUs operate fastest when threads on a multiprocessor perform the same calculation, NAMD pre-sorts atomic coordinates before transferring them to the GPU to reduce the likelihood that threads on a multiprocessor

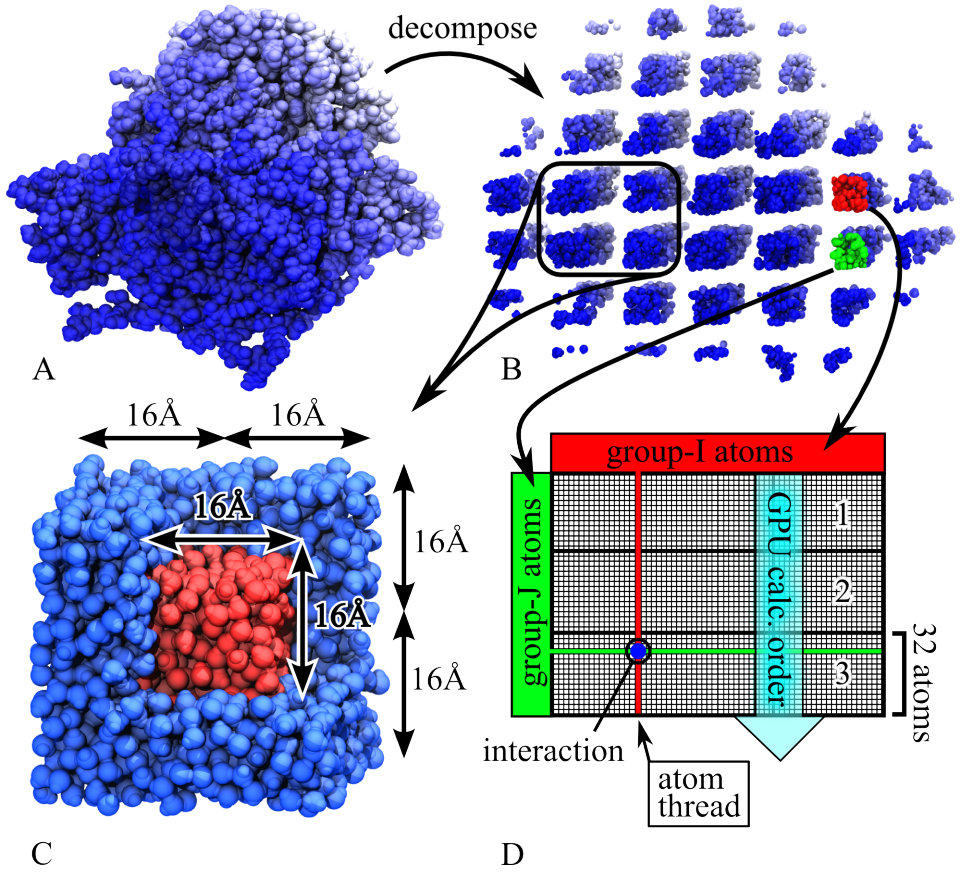

Figure 4.1: Hybrid GB/SA decomposition. (A) Simulated protein glycogen phosphorylase. (B) Protein decomposed into a 3D grid of atom-groups. (C) LCPO force work unit involving a $2 \times 2 \times 2$ set of 8 adjacent atom-groups. The force work unit calculates forces for the inner 1/8th core of atoms (red) due to overlap with neighbors (red and blue). (D) Thread block design for GB force calculation on GPU multiprocessor. Each thread (vertical line) calculates force on one group-I atom (red) due to interactions (blue) with group-J atoms (green); group-J atoms are loaded 32 at a time, in order 1-3, for coalesced and, therefore, accelerated memory access.

must evaluate differing pairwise functions, $f_{1-4,i}$ of Eq. 4.9, as doing so dramatically reduces performance.

4.2.6 SA Force Calculation on CPUs

Because Eq. 4.13 requires iteration over all atom triplets, the LCPO calculation cannot be carried out using the structure of atom-group pairs which NAMD employs for Coulomb, van der Waals and GB force calculation, as it would be possible for a triplet of three overlapping atoms to belong to three neighboring atom-groups, in which case no atom-group pair would evaluate the triplet. Therefore, an LCPO force work unit instead consists of a $2 \times 2 \times 2$ set of 8 adjacent atom-groups as illustrated in Fig. 4.1C. Each LCPO force work unit calculates the surface area and hydrophobic force on the inner $1/8^{\text{th}}$ fraction of atoms (Fig. 4.1C red) due to surface area overlaps with neighboring atoms (red and blue). LCPO force work units can also

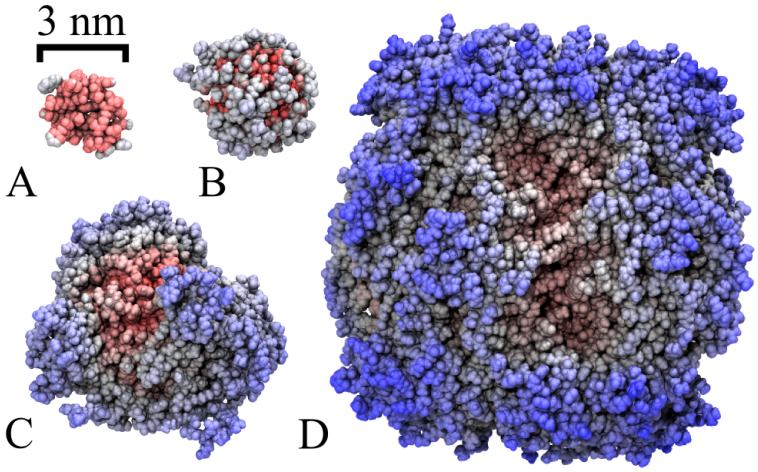

Figure 4.2: Benchmarked protein systems. (A) Villin headpiece (PDB ID 1YRI) with 582 atoms; (B) flavodoxin (PDB ID 2W5U) with 2,412 atoms; (C) glycogen phosphorylase (PDB ID 1GPB) with 13,340 atoms; (D) RuBisCO (PDB ID 1IR2) with 74,926 atoms.

be partitioned such that the force work unit in Fig. 4.1C is duplicated on multiple CPU cores with each core calculating forces for only a fraction of the inner core atoms (red). The ability to partition the SA calculation into many small force work units will be shown to be critical to the efficiency of the hybrid GPU/CPU algorithm.

4.2.7 Balancing the GPU and CPU Calculations

With fast algorithms in place to perform the GB force calculation on the GPU and the SA force calculation on the CPU, the remaining obstacle of combining the two into a hybrid GB/SA calculation requires coordination of the GPU and CPU computation. First, the right balance of GPUs and CPU cores will allow the Coulomb, van der Waals and GB calculation on the GPU and the SA calculation on the CPU to be executed in the same amount of time. Second, partitioning the SA calculation into small force work units will allow the host CPU to switch between SA calculations and GPU interaction with high frequency, thereby minimizing the delay of either calculation. While thorough benchmarking of our implementation will judge performance, examining the calculation process will additionally verify the ideal ratio of GPUs to CPU cores for efficient performance, as well as verify the simultaneous overlap of the two calculations.

4.2.8 Simulations Carried Out

To analyze performance of NAMD's hybrid GB/SA algorithm, four test systems differing in size were simulated on 0-4 GPUs and 1-32 CPU cores. Fig. 4.2 depicts the four tested systems and Table 4.1 lists their

Name	PDB ID	# Atoms	NAMD SA	Amber SA
villin headpiece	1YRI	582	2,706.72	2,706.72
flavodoxin	2W5U	2,412	9,988.72	9,988.72
glycogen phosphorylase	1GPB	13,340	46,106.66	46,106.68
RuBisCO	1IR2	74,926	176,335.86	176,335.90

Table 4.1: Benchmarked protein systems. Listed are associated surface areas, in units Å^2, as calculated by NAMD's and Amber's implementations of the LCPO [54] algorithm.

SA values; hydrogen atoms missing in the PDB structure files were placed by VMD's [81] psfgen tool.

For benchmarking, three types of simulations were performed, each with increasing computational cost and solvent accuracy. In vacuo simulations evaluate only the Coulomb and van der Waals non-bonded forces as defined through Eqs. 4.1-4.3, GB simulations additionally evaluate the generalized Born forces, determined through Eqs. 4.4-4.11, and GB/SA simulations further include the SA calculation following the LCPO algorithm as stated through Eqs. 4.12-4.14.

The following simulation parameters were employed for all simulations. A value of 16 Å was employed for the Coulomb and van der Waals interaction cut-off as well as for the GB phase 2 calculation, c.f. Eq. 4.11, while the GB phase 1 and 3 calculations, as defined through Eqs. 4.8 and 4.10, were cut off at 14 Å. For GB and GB/SA simulations, an implicit ion concentration of 0.3 M was assumed [108]. The GB/SA simulations employed a surface tension of 0.005 kcal/mol/Å^2 [156] unless otherwise specified. A time step of 2 fs was employed, requiring the SHAKE [157] algorithm to restrain covalent bonds to hydrogen atoms.

Benchmark simulations were run for 620 time steps on a 2.2 GHz multi-core desktop computer accelerated with NVIDIA Tesla T20 GPUs. The first 20 steps of simulation perform conjugate gradient minimization; the next 500 steps consisted of NAMD load balancing to optimize parallel performance; the simulation speed, in units seconds/time step, was averaged over the final 100 steps. Results of the benchmark simulations are presented in Fig. 4.3 and 4.4.

To verify the computational accuracy of NAMD's new GPU-accelerated GB calculation and multi-core LCPO calculation, energy and surface area calculations were compared to those of prior reference implementations [43, 155]. The relative error, $(E_{\text{ref}} - E_{\text{new}})/E_{\text{ref}}$, of NAMD's GB energy calculation, determined through Eq. 4.4, on the GPU, with NAMD's CPU implementation [43] as reference, is less than 5×10^{-6} for all four benchmark proteins. The relative error, $(\text{SA}_{\text{ref}} - \text{SA}_{\text{new}})/\text{SA}_{\text{ref}}$, of NAMD's surface area calculation, determined through Eq. 4.13, with the Amber [155] implementation as reference, is less than 4×10^{-7} for all four benchmark systems; total molecular surface areas of the four systems, as calculated by the Amber and NAMD implementations of LCPO, are listed in Table 4.1.

Hydrophobic solvation energy is a significant contributor to implicit solvent behavior [158], necessitating

it's inclusion with GB calculations. The effect of the hydrophobic energy contributions, accounted for in the SA calculation, was explored by simulating the 2W5U benchmark system using the GB/SA implicit solvent, employing eight different surface tension parameter values. Additionally, the 2W5U system was simulated in TIP3P [83] explicit solvent to determine which surface tension parameter values most closely reproduce protein behavior in explicit solvent. With the experimentally measured surface tension of hydrocarbons in aqueous solvent being 0.005 kcal/mol/Å^2 [156], surface tension parameter values in the range 0.001-0.128 kcal/mol/Å^2 were tested. Using otherwise the same simulation parameters previously outlined, the 2W5U system was equilibrated for 1.0 ns, at which time the total surface area, a molecular property closely affected by surface tension, had reached an equilibrium value. The surface areas evaluated during the simulations are plotted in Fig. 4.5; for the TIP3P simulation, only the final protein surface area, as calculated by VMD's [81] high precision solvent-accessible surface area tool, with a solvent probe radius of 1.4 Å, is shown.

4.3 Results

The performed simulations examine the hybrid algorithm by verifying simultaneous overlap of GPU and CPU calculations, estimating the ideal ratio of GPUs to CPU cores which maximizes computing efficiency, benchmarking performance and evaluating appropriate values for the surface tension parameter.

4.3.1 Hybrid GB/SA Performance Analysis

Central to fast hybrid GPU/CPU algorithms is the simultaneous overlap of calculations on both processor types, brought about by the host CPUs coordinating the GB calculation on the GPU with the SA calculations on the CPU. To demonstrate this coordination, Fig. 4.3 demonstrates, using the Projections [159] tool, the execution process of a GB/SA simulation of the 1GPB system on multiple GPUs and CPU cores.

Fig. 4.3A illustrates the GB/SA calculation executed on 16 CPU cores and no GPUs. The three phases of the GB calculation, described by Eqs. 4.8, 4.11 and 4.10, are carried out with SA calculations interspersed to utilize the CPU when it would otherwise be idle while waiting for other cores to complete the phase; the lack of idle time (shown in yellow) signifies a highly efficient calculation.

Fig. 4.3B demonstrates how the GB/SA calculation executed solely on 16 CPU cores is accelerated through the addition of 1 GPU. Because of its powerful computing capability, the GPU completes all Coulomb, van der Waals and GB calculations in the same time that the 16 CPU cores need to perform the SA calculation, resulting in a three-fold overall performance increase.

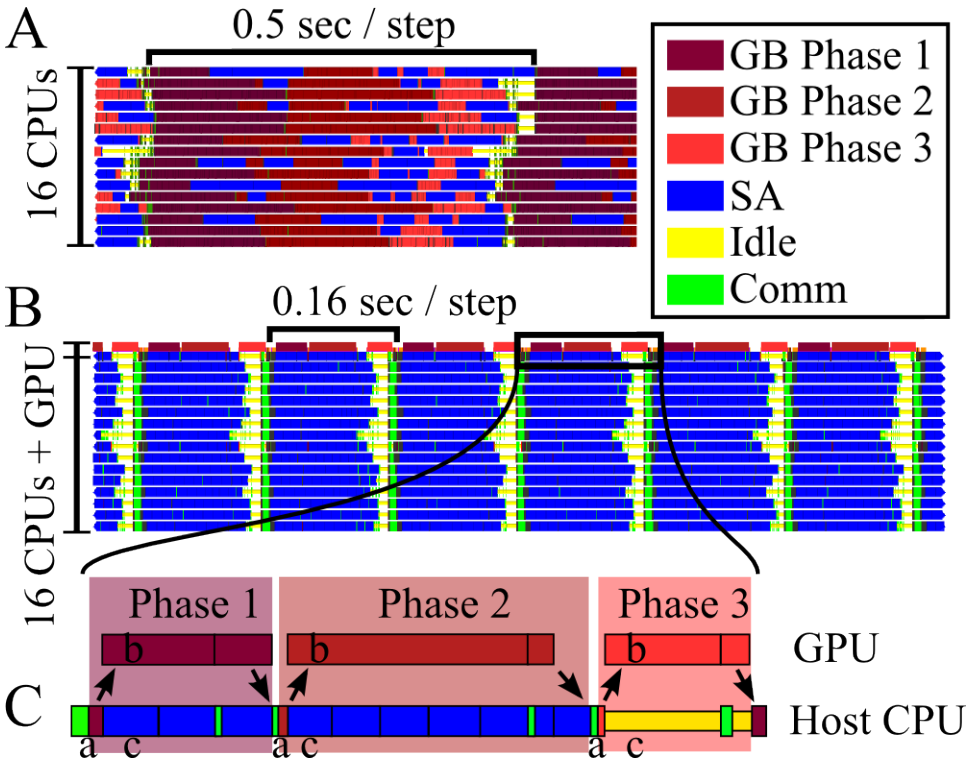

Figure 4.3: NAMD's GB/SA calculation. (A) Performance on 16 CPUs only. (B) Performance on 16 CPUs with 1 GPU added. Colors represent calculations being performed on the processors: three phases of GB calculation (see text); SA calculation; idle time; communication. (C) Detailed view of host CPU switching between communicating with the GPU and performing the SA calculation; arrows represent the transfer of data between host CPU and GPU; for GB phase 1, 2 and 3 calculations: (a) host CPU initializes the GB calculation on the GPU; (b) GPU calculates a GB phase while (c) host CPU performs SA calculations.

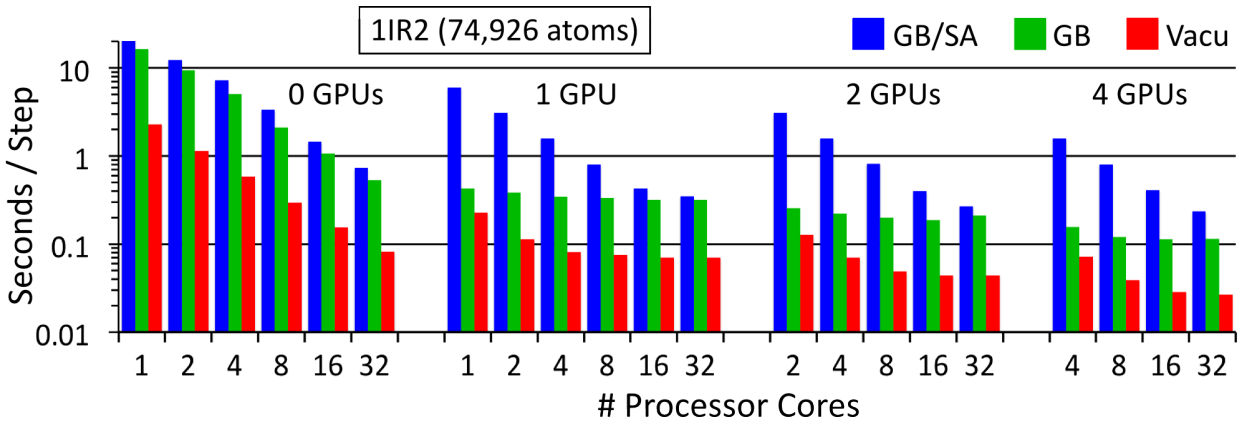

Figure 4.4: Simulation speed, in seconds / time step, for the 1IR2 benchmark system on 0-4 GPUs and 1-32 CPU cores; speeds for in vacuo (red), GB (green) and GB/SA (blue) calculations are shown.

A bottleneck for hybrid GPU/CPU calculations is the CPU's limited ability to switch between performing the SA calculation and GPU-related operations. Fig. 4.3C illustrates the interplay between the GPU execution and the host CPU calculations. For each of the three GB phases, the host CPU initializes the GB calculation on the GPU then, while the GPU calculates the GB phase, the host CPU performs SA calculations. NAMD efficiently overlaps the GB and SA calculations by partitioning the SA calculation into many short, independent force work units, such that at the completion of each SA force work unit, the CPU can engage in GPU-related operations, if needed, then return to the next SA force work unit as highlighted by Fig. 4.3C.

Because of the performance difference between GPU and CPU and the different computational cost of the GB and SA calculations, it was not known what ratio of GPUs to CPUs would be most efficient. The best ratio has GPUs and CPUs requiring the same time to perform their respective force calculations. Based on the data in Table 4.2 and as illustrated in Fig. 4.3, a ratio of 16 CPU cores per GPU allowed highly efficient performance, while deviating from this ratio resulted in either the GB or SA calculation to become rate limiting. As both GPU and CPU technologies evolve, the ideal ratio of CPU cores per GPU will also evolve, both for our GB/SA algorithm as well as for other hybrid GPU/CPU algorithms.

4.3.2 Performance Benchmark Results

The performance of the hybrid GB/SA implementation is demonstrated by the benchmarks, which are reported in full in Table 4.2 and illustrated for the 1IR2 system in Fig. 4.4. The incorporation of GPUs into the MD simulation computation offers dramatic performance increases; for example, for the GB simulations of the 1IR2 system, simulations on 1 GPU are 25% faster than on 32 CPU cores. Additionally, the three largest systems tested were simulated, with the GB model, 30-35 times faster on 1 GPU than on 1 CPU

		1YRI			2W5U			1GPB			1IR2		
GPU	CPU	Vacu	GB	GB/SA	Vacu	GB	GB/SA	Vacu	GB	GB/SA	Vacu	GB	GB/SA
0	1	8.4E-3	4.9E-2	7.3E-2	4.9E-2	3.5E-1	5.2E-1	3.4E-1	2.4E+0	3.4E+0	2.3E+0	1.6E+1	2.2E+1
	2	5.7E-3	2.7E-2	4.0E-2	2.7E-2	1.8E-1	2.6E-1	1.8E-1	1.2E+0	1.7E+0	1.2E+0	9.5E+0	1.2E+1
	4	3.8E-3	1.5E-2	2.2E-2	1.4E-2	9.3E-2	1.3E-1	9.4E-2	6.2E-1	8.8E-1	5.9E-1	5.1E+0	7.2E+0
	8	2.6E-3	8.7E-3	1.3E-2	8.0E-3	4.7E-2	6.8E-2	4.8E-2	3.2E-1	4.4E-1	3.0E-1	2.1E+0	3.4E+0
	16	2.2E-3	5.6E-3	7.5E-3	4.8E-3	2.6E-2	3.7E-2	2.6E-2	1.6E-1	2.3E-1	1.6E-1	1.1E+0	1.5E+0
	32	1.7E-3	3.7E-3	4.9E-3	3.1E-3	1.4E-2	2.1E-2	1.4E-2	8.3E-2	1.2E-1	8.2E-2	5.4E-1	7.3E-1
1	1	3.6E-3	3.8E-3	3.1E-2	8.1E-3	1.1E-2	1.7E-1	4.3E-2	6.7E-2	1.0E+0	2.3E-1	4.3E-1	6.0E+0
	2	3.2E-3	4.4E-3	1.9E-2	5.0E-3	1.1E-2	8.9E-2	2.3E-2	5.8E-2	5.2E-1	1.1E-1	3.9E-1	3.1E+0
	4	2.1E-3	3.4E-3	1.2E-2	3.6E-3	9.9E-3	4.6E-2	1.4E-2	5.4E-2	2.7E-1	8.1E-2	3.5E-1	1.6E+0
	8	1.8E-3	3.3E-3	6.9E-3	2.8E-3	1.0E-2	2.7E-2	1.3E-2	5.1E-2	1.4E-1	7.5E-2	3.3E-1	8.0E-1
	16	2.1E-3	3.3E-3	4.8E-3	2.7E-3	1.1E-2	1.8E-2	1.3E-2	5.4E-2	8.1E-2	7.1E-2	3.2E-1	4.3E-1
	32				2.8E-3	8.9E-3	1.2E-2	1.2E-2	5.0E-2	6.1E-2	7.0E-2	3.2E-1	3.5E-1
2	2	3.0E-3	3.9E-3	2.0E-2	5.0E-3	9.2E-3	8.9E-2	2.4E-2	4.4E-2	5.2E-1	1.3E-1	2.6E-1	3.1E+0
	4	2.3E-3	3.4E-3	1.1E-2	3.7E-3	9.2E-3	5.0E-2	1.4E-2	3.7E-2	2.7E-1	7.1E-2	2.2E-1	1.6E+0
	8	1.8E-3	3.2E-3	7.9E-3	2.5E-3	8.6E-3	2.9E-2	1.0E-2	3.6E-2	1.4E-1	4.9E-2	2.0E-1	8.1E-1
	16	1.7E-3	3.2E-3	5.8E-3	2.7E-3	8.7E-3	1.9E-2	8.9E-3	3.8E-2	8.1E-2	4.4E-2	1.9E-1	4.0E-1
	32				2.5E-3	7.6E-3	1.3E-2	7.9E-3	3.4E-2	5.3E-2	4.4E-2	2.1E-1	2.7E-1
4	4	2.7E-3	3.4E-3	1.2E-2	3.5E-3	1.0E-2	5.0E-2	1.4E-2	3.0E-2	2.7E-1	7.2E-2	1.6E-1	1.6E+0
	8	1.8E-3	3.1E-3	1.2E-2	2.5E-3	8.1E-3	3.0E-2	8.5E-3	2.6E-2	1.4E-1	3.9E-2	1.2E-1	8.0E-1
	16	2.2E-3	3.2E-3	6.3E-3	2.6E-3	7.9E-3	2.1E-2	7.2E-3	2.8E-2	9.1E-2	2.9E-2	1.1E-1	4.1E-1
	32				2.4E-3	6.7E-3	1.2E-2	6.4E-3	2.3E-2	4.7E-2	2.7E-2	1.2E-1	2.4E-1

Table 4.2: Benchmark simulation speeds, in units seconds / time step, for the four benchmarked protein systems tested on 0-4 GPUs and 1-32 CPU cores. Speeds are reported for three increasingly accurate and expensive simulation types: in vacuo (Vacu), GB, and GB/SA. The small size of system 1YRI, see Fig. 4.2A, prohibited utilization of 32 CPU cores in some cases.

core, while even the smallest system, 1YRI, was simulated 13 times faster on 1 GPU than on 1 CPU core. The GB/SA simulation of the 1IR2 system executed almost 50 times faster on 1 GPU with 16 CPU cores than on 1 CPU core alone. The impressive speed-ups achieved by utilizing GPU technology demonstrate the benefit which GPUs offer to MD computing. While the benchmarks reported here are only for up to 4 GPUs, the hybrid algorithm also operates on supercomputers built from many GPUs and multi-core CPUs as previously described for GPU-acceleration of NAMD [147].

4.3.3 Testing Surface Tension Parameters

When employing the SA calculation, the surface tension, in units kcal/mol/Å^2, is a parameter of the model, controlling hydrophobic energy of solvation; is not a physical property measured from the simulation. To explore the effect of the parameter on a protein system, the 2W5U system was equilibrated through a GB/SA implicit solvent simulation employing surface tension parameters ranging from 0.001 to 0.128 kcal/mol/Å^2; the system was also equilibrated through a standard all-atom simulation employing explicit TIP3P [83] solvent. Fig. 4.5 presents protein surface area arising in GB/SA equilibration simulations as well as the final surface area of each. The protein surface area is expected to diminish with increasing surface tension as the latter imparts an energy penalty for the protein's surface exposed to solvent. The simulations utilizing surface tension parameter values of 0.004 and 0.008 kcal/mol/Å^2 returned final surface areas closest to the area of the reference TIP3P equilibrated system, suggesting that employing surface tension parameter values between 0.004 and 0.008 kcal/mol/Å^2 most closely reproduce protein behavior in explicit solvent. The experimentally measured surface tension of hydrocarbons in aqueous solvent is 0.005 kcal/mol/Å^2 [156] which validates our finding.

4.4 Summary

The structural biology community has been well served by technological advances of general purpose GPU computing, which have made molecular dynamics more powerful and accurate. Already, many biological computing programs have achieved great improvements in performance through GPU acceleration. As it is often neither feasible nor ideal to perform all needed calculations on the GPU, it becomes increasingly important to develop methods for overlapping GPU and CPU calculations. From the present work results an efficient method for performing GB/SA simulations on hybrid GPU/CPU computers, permitting extremely fast and accurate molecular dynamics simulations that can be executed, for example, interactively by researchers.

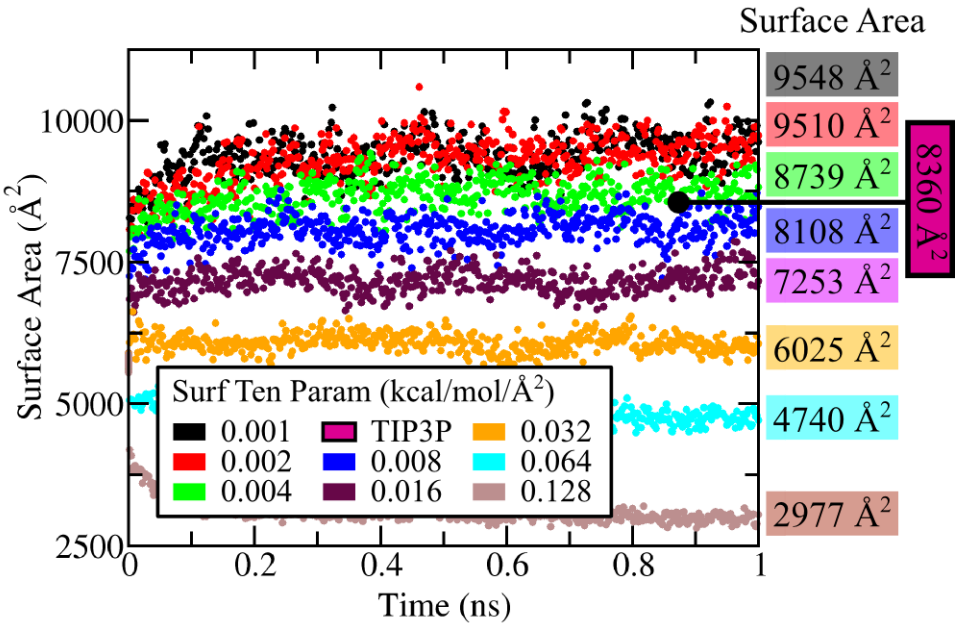

Figure 4.5: Surface area during eight GB/SA simulations of the 2W5U system employing different surface tension parameter values. The final surface area for each GB/SA simulation is listed at right; shown is also the final surface area resulting from the simulation with explicit (TIP3P) solvent.

Chapter 5

Conclusions

The molecular dynamics (MD) methodology has proven a vital tool for exploring the dynamics of molecular biological systems [42]. The utility of MD is expanded as the reachable timescales lengthen to encompass cellular processes. The three examples discussed in this work, the theoretical model, implicit solvent, and advanced computing technologies, represent methods for lengthening the timescale reach of molecular dynamics. Fig. 5.1 illustrates the degree to which each of the three avenues may affect MD simulations.

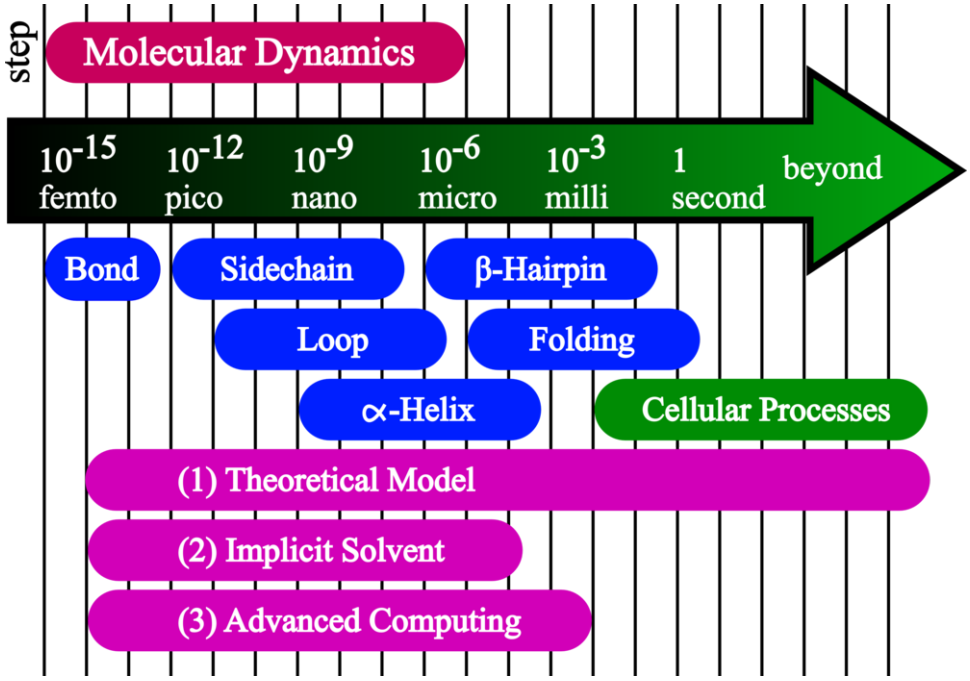

Figure 5.1: With the use of theoretical models, implicit solvent, and advanced computing technologies, molecular dynamics simulations are able to reach longer timescales. The theoretical model of flagellin translocation enables MD simulations to explore the hour-long process of flagellum elongation. The use of implicit solvent, over explicit solvent, delivers a 10X timescale improvement, due to faster simulation speed and accelerated conformational sampling, as demonstrated by the MDFF simulation of the ribosome. The employment of GPUs for implicit solvent simulations can increase simulation speeds by 10-100X versus a single CPU core.

The model of flagellin translocation links local and short time properties to overall translocation, allowing the compression simulations, which cover only nanometers in size and nanoseconds in time, to represent the

growing flagellum, which grows several micrometers in length over several hours. Such theoretical models [42] are vital for applying MD simulations to cellular timescale biological processes.

The generalized Born implicit solvent (GBIS) model has long been employed for molecular dynamics simulations of relatively small bio-molecules. NAMD's unique GBIS implementation can also simulate very large systems, such as the entire ribosome, and does so efficiently on large parallel computers. The new GBIS capability of NAMD will be beneficial to accelerate in simulations the slow motions common to large systems by eliminating viscous drag from water, thus improving feasibility of 10X longer timescale simulations.

The structural biology community has been well served by technological advances which have made more powerful the computational microscope of molecular dynamics. Already, many biological computing programs have achieved great improvements in performance through GPU acceleration. As it often not feasible nor ideal to perform every necessary calculation on the GPU, it becomes increasingly important to develop methods for overlapping GPU and CPU calculations as such will achieve optimal performance. The GPU-acceleration work demonstrated an efficient method for performing GB/SA simulations on hybrid GPU/CPU computers, making NAMD a highly effective tool for hybrid structural calculations such as GB/SA. By applying GPU technology to implicit solvent simulations, MD achieves speedups of 10-100X.

The three methods here discussed, combine to lengthen the timescale reach of molecular dynamics simulations several orders of magnitude, which brings closer the possibility of computational simulations of cellular processes.

Appendix A

Pressure - Density Relationship of a Gaussian Chain Polymer Confined to a Cylinder

In the following we seek to show that an exponent $\beta \approx 3$ results in the pressure - density relationship $p = \gamma\rho^{\beta}$ for a polymer confined to a cylinder of radius a and length L. The confinement applies to the unfolded flagellin in the flagellar channel as studied in the present paper. In our description we consider the protein to be a linear polymer, i.e., we neglect the detailed structure of the amino acid side chains.

The polymer property is accounted for through a so-called Gaussian chain model [77, 78]. In this model a polymer consists of N segments represented through vectors $\mathbf{r}_j$, $j = 1, 2, \ldots N$, such that the vector connecting the beginning with the end of the polymer is given by $\sum_{j=1}^{N} \mathbf{r}_j$. The $\mathbf{r}_j$ are distributed isotropically according to a Gaussian distribution

$$\phi(\mathbf{r}_j) = \left[3/2\pi b^2\right]^{\frac{3}{2}} \exp\left[-3r_j^2/2b^2\right] . \tag{A.1}$$

Since the distributions $\phi(\mathbf{r}_j)$ are independent of each other, the polymer can intersect with itself. The Gaussian model, therefore, does not apply well to a description where each segment described through $\mathbf{r}_j$ is literally a polymer segment, but rather applies best, though still only very approximately, when the $\mathbf{r}_j$ represent large polymer units with a length $|\mathbf{r}_j| = b$; b is suggested to be two times the so-called persistence length of the polymer [78] (see Sect. 2.6.2). The persistence length is the polymer correlation length characteristic of the directional order; beyond the persistence length, directional correlation of the chain is lost.

The Gaussian chain polymer behaves like an entropic spring governed by the Hamiltonian [78] (see Sect. 2.3)

$$H_0 = \frac{3k_{\mathrm{B}}T}{2b^2} \sum_{i=1}^{N} |\mathbf{R}_{i+1} - \mathbf{R}_i|^2 . \tag{A.2}$$

This behavior follows from the fact that for a polymer with independent segments distributed according to

Material in this appendix, contributed mainly from Wen Ma, is reproduced in part with permission from David E. Tanner, Wen Ma, Zhongzhou Chen and Klaus Schulten, "Theoretical and computational investigation of flagellin translocation and bacterial flagellum growth," *Biophysical Journal*, 100:607-614 (2011).

Eq. A.1 any polymer piece involving segments $j, j+1, \ldots i-1$ has an end-end distribution [77] (see Chap. II)

$$\phi(\mathbf{R}_i \, - \, \mathbf{R}_j, i \, - \, j) \; = \; \left[3/2\pi b^2 |i \, - \, j|\right]^{3/2} \exp\left[-\, 3\,(\mathbf{R}_i \, - \mathbf{R}_j)^2/2|i \, - \, j|b^2\right] \; . \tag{A.3}$$

Here $\mathbf{R}_i$ denotes the starting point of segment i.

Our strategy to determine the pressure - density relationship for the present polymer model will be to exploit the relationship

$$p \; = \; -\,\frac{1}{\pi a^2}\frac{dF}{dL} \; . \tag{A.4}$$

Here F is the free energy of the system related to the partition function Z through $F \, = \, -k_{\rm B}T \, \ln Z$. Z can be expressed through the end-end distribution function $\rho(\mathbf{R}, \mathbf{R}', N)$ where $\mathbf{R}'$ is the starting point of the polymer, $\mathbf{R}$ is the end point of the polymer, and N denotes the number of polymer segments

$$Z \; = \; \int\int d\mathbf{R}\, d\mathbf{R}' \; \rho(\mathbf{R}, \mathbf{R}', N) \; . \tag{A.5}$$

The polymer described in Eq. A.4 and Eq. A.5 has to be confined to the cylinder of radius a and length L. However, the polymer described through Eq. A.1, Eq. A.2 and Eq. A.3 is not confined. In order to obtain $\rho(\mathbf{R}, \mathbf{R}', N)$ for a confined polymer we apply, following [78], to the unconfined polymer a confining potential $V(\mathbf{R}_j)$, to be specified further below. For this purpose we modify the Hamiltonian H_0 to

$$H \; = \; \frac{3k_{\rm B}T}{2b^2} \sum_{i=1}^{N} |\mathbf{R}_{i+1} \, - \, \mathbf{R}_i|^2 \; + \; \sum_{j=0}^{N+1} V(\mathbf{R}_j) \; . \tag{A.6}$$

The sum over segments is extended from N to $N+1$ to include in the potential the term $V(\mathbf{R}_{N+1})$ for the end point of the polymer. The polymer subjected to the potential $V(\mathbf{R}_j)$ exhibits a distribution of its conformations $\{\mathbf{R}_1, \mathbf{R}_2, \ldots \mathbf{R}_{N+1}\}$

$$Q(\mathbf{R}_1, \mathbf{R}_2, \ldots \mathbf{R}_{N+1}) \; \sim \; \exp\left[-\,\frac{3}{2b^2} \sum_{i=1}^{N} |\mathbf{R}_{i+1} \, - \, \mathbf{R}_i|^2 \; - \; \sum_{j=0}^{N+1} \frac{1}{k_{\rm B}T} V(\mathbf{R}_j)\right] \tag{A.7}$$

The corresponding end-end distribution function $\rho(\mathbf{R}, \mathbf{R}', N)$, where $\mathbf{R}_1 \, = \, \mathbf{R}$ and $\mathbf{R}_{N+1} \, = \, \mathbf{R}'$, is obtained by averaging over the intermediate polymer bead positions $\{\mathbf{R}_2, \mathbf{R}_3, \ldots \mathbf{R}_N\}$, i.e.,

$$\rho(\mathbf{R}, \mathbf{R}', N) \; = \; \int d\mathbf{R}_2 \int d\mathbf{R}_3 \, \cdots \int d\mathbf{R}_N \; Q(\mathbf{R}, \mathbf{R}_2, \ldots \mathbf{R}_N, \mathbf{R}') \; . \tag{A.8}$$

Using Eq. A.7 this can be written symbolically

$$\rho(\mathbf{R}=\mathbf{R}_1,\mathbf{R}'=\mathbf{R}_{N+1},N) \sim \int d\mathbf{R}_2 \int d\mathbf{R}_3 \cdots \int d\mathbf{R}_N \times$$
$$\times \exp\left[-\frac{3}{2b^2}\sum_{i=1}^{N}|\mathbf{R}_{i+1}-\mathbf{R}_i|^2 - \sum_{j=0}^{N+1}\frac{1}{k_\mathrm{B}T}V(\mathbf{R}_j)\right] \tag{A.9}$$

Carrying out this calculation for $V \equiv 0$ yields

$$\rho_0(\mathbf{R},\mathbf{R}',N) = \left[3/2\pi b^2 N\right]^{3/2}\exp\left[-3\,(\mathbf{R}-\mathbf{R}')^2/2Nb^2\right] \tag{A.10}$$

which agrees, as expected, with the unconfined polymer distribution function in Eq. A.3.

In order to determine $\rho(\mathbf{R},\mathbf{R}',N)$ for an external potential $V(\mathbf{R}_j)$ we do not carry out the integration in Eq. A.9, but rather follow a procedure suggested in [78] (see Sect. 3.2) that determines $\rho(\mathbf{R},\mathbf{R}',N)$ as a solution of the partial differential equation

$$\left[\frac{\partial}{\partial N} - \frac{b^2}{6}\nabla^2 + \frac{1}{k_\mathrm{B}T}V(\mathbf{R})\right]\rho(\mathbf{R},\mathbf{R}',N) = 0 \tag{A.11}$$

where ∇^2 is the Laplacian with respect to $\mathbf{R}$. This solution must obey the condition

$$\rho(\mathbf{R},\mathbf{R}',0) = \delta(\mathbf{R}-\mathbf{R}')\,. \tag{A.12}$$

In the present case, the confining potential for a cylinder is defined through

$$V(\mathbf{R}) = \begin{cases} 0 & \mathbf{R} \text{ inside cylinder} \\ \infty & \mathbf{R} \text{ outside cylinder} \end{cases}\,. \tag{A.13}$$

One can conclude readily from this definition that $\rho(\mathbf{R},\mathbf{R}',N)$ should obey then

$$\left[\frac{\partial}{\partial N} - \frac{b^2}{6}\nabla^2\right]\rho(\mathbf{R},\mathbf{R}',N) = 0 \tag{A.14}$$

and vanish for $\mathbf{R}$ and $\mathbf{R}'$ on the cylinder surface. It should also obey condition Eq. A.12.

Equation A.14 with $N = t$ (time) and $b^2/6 = D$ (diffusion coefficient), is equivalent to the free diffusion equation, solutions of which are well known. In fact, the solution of Eq. A.14 for a cylindrical boundary condition can be found in [160] (see Sect. 8.6). According to Eq. A.5 we need to determine actually the

quantity

$$\sigma(\mathbf{R}, N) \;=\; \int d\mathbf{R}'\, \rho(\mathbf{R}, \mathbf{R}', N)\;. \tag{A.15}$$

In the case of axial symmetry, that applies in the present case, one obtains the solution as stated in [160] after some simple algebra

$$\sigma(\mathbf{R}, N) \;=\; \sum_{\mu} \sum_{m=1}^{\infty} A_{\mu m}\, J_0(\mu r)\, \sin[m\pi z/L]\, \exp[-D\{\mu^2 \,+\, (m^2\pi^2/L^2)\}\, N] \tag{A.16}$$

where we used the cylindrical coordinates $\mathbf{R} \;=\; (r, \phi, z)$ (ϕ drops out due to axial symmetry). The expansion coefficients $A_{\mu m}$ are

$$A_{\mu m} \;=\; \frac{8}{a^2 L[J_1(\mu a)]^2} \int_0^a r\, J_0(\mu r)\, dr \int_0^L \sin[m\pi z/L] dz\;. \tag{A.17}$$

In Eq. A.16 and Eq. A.17, J_0 and J_1 are the cylindrical Bessel functions of zeroth and first order; the summation index μ denotes the zeroes of $J_0(\mu a)$. Using

$$\int_0^a r\, J_0(\mu r)\, dr \;=\; \frac{a}{\mu}\, J_1(\mu a) \tag{A.18}$$

and

$$\int_0^L \sin[m\pi z/L] dz \;=\; \frac{L}{m\pi}[1 \,-\, (-1)^m] \tag{A.19}$$

one obtains finally

$$A_{\mu m} \;=\; \frac{8\,[1 - (-1)^m]}{a\,\mu\,m\,\pi\,[J_1(\mu a)]}\;. \tag{A.20}$$

We need to determine now (c.f. Eq. A.15 and Eq. A.5)

$$Z \;=\; \int d\mathbf{R}\, \sigma(\mathbf{R}, N)\;. \tag{A.21}$$

where $\sigma(\mathbf{R}, N)$ is given by Eq. A.16. Using again the integral values from Eq. A.18 and Eq. A.19 one obtains

$$Z \;=\; \sum_{\mu} \sum_{m=1,3,5\ldots} \frac{64L}{m^2\mu^2\pi}\, \exp[-D\{\mu^2 \,+\, (m^2\pi^2/L^2)\}\, N]\;. \tag{A.22}$$

According to Eq. A.4 the pressure is

$$p \;=\; \frac{k_{\mathrm{B}}T}{\pi a^2} \frac{\partial \ln Z}{\partial L}\;. \tag{A.23}$$

Factoring Z into an L-independent and an L-dependent term

$$Z \;=\; Z_1\, Z_2(L) \tag{A.24}$$

where

$$Z_1 \;=\; \sum_{\mu} \frac{64}{\mu^2 \pi} \exp(-D\,\mu^2\, N) \tag{A.25}$$

and

$$Z_2(L) \;=\; L \sum_{m=1,3,5\ldots} \frac{1}{m^2} \exp(-\,D\,N m^2 \pi^2 / L^2) \tag{A.26}$$

one can write

$$p \;=\; \frac{k_{\rm B} T}{\pi a^2} \frac{1}{Z_2(L)} \frac{\partial Z_2(L)}{\partial L}\;. \tag{A.27}$$

From this results

$$p \;=\; \frac{k_{\rm B} T}{\pi a^2} \left\{ \frac{1}{L} \;+\; \frac{N \pi^2 b^2}{3 L^3} \frac{\sum_{m=1,3,5\ldots} \exp(-b^2 \pi^2 N m^2 / 6 L^2)}{\sum_{n=1,3,5\ldots} (1/n^2)\, \exp(-b^2 \pi^2 N n^2 / 6 L^2)} \right\}\;. \tag{A.28}$$

One can recognize that the two series arising in this expression converge quickly for $\alpha \;=\; 3 N \pi^2 b^2 / 2 L^2 \;>>\; 1$. In the case of the system simulated in the present case typical α values obey $\alpha \;>\; 15$ and, hence, one may limit the two series after the first term, yielding

$$p \;=\; \frac{k_{\rm B} T}{\pi a^2} \left[\frac{1}{L} \;+\; \frac{N \pi^2 b^2}{3 L^3} \right]\;. \tag{A.29}$$

In order to compare Eq. A.29 to the simulation data in Fig. 5 we employ the expression for the atomic density

$$\rho_{\rm atom} \;=\; N_{\rm atom} / \pi a^2 L \tag{A.30}$$

which permits one to express

$$L \;=\; N_{\rm atom} / \pi a^2 \rho_{\rm atom}\;. \tag{A.31}$$

One can also express the number of polymer segments N by L_0, the length of the totally stretched polymer main chain,

$$N \;=\; L_0 / b\;. \tag{A.32}$$

Employing Eq. A.31 and Eq. A.32 in Eq. A.29 results in

$$p \;=\; k_{\mathrm{B}}T \left[\frac{\rho_{\mathrm{atom}}}{N_{\mathrm{atom}}} \;+\; \frac{\pi^4 a^4 b\, L_0}{3} \left(\frac{\rho_{\mathrm{atom}}}{N_{\mathrm{atom}}} \right)^3 \right] \;. \tag{A.33}$$

Introducing

$$\tilde{\rho} \;=\; \rho_{\mathrm{atom}}/N_{\mathrm{atom}} \;, \tag{A.34}$$

which stands for the density of one polymer chain, one obtains finally the pressure - density relationship for the Gaussian chain model polymer confined to a cylinder

$$p \;=\; k_{\mathrm{B}}T \left[\tilde{\rho} \;+\; \frac{1}{3}\pi^4 a^4 b\, L_0\, \tilde{\rho}^3 \right] \;. \tag{A.35}$$

For physical parameters typical in the flagellum, the cubic term of Eq. A.35 is one to two orders of magnitude larger than the linear term; pressure, therefore, is approximately proportional to the cube of the density, i.e., $p \;\propto\; \tilde{\rho}^3$.

References

[1] Lee, E. H., J. Hsin, M. Sotomayor, G. Comellas, and K. Schulten. 2009. Discovery through the computational microscope. *Structure.* 17:1295–1306.

[2] Freddolino, P. L. and K. Schulten. 2009. Common structural transitions in explicit-solvent simulations of villin headpiece folding. *Biophys. J.* 97:2338–2347.

[3] Trabuco, L. G., E. Villa, E. Schreiner, C. B. Harrison, and K. Schulten. 2009. Molecular Dynamics Flexible Fitting: A practical guide to combine cryo-electron microscopy and X-ray crystallography. *Methods.* 49:174–180.

[4] Freddolino, P. L., A. S. Arkhipov, S. B. Larson, A. McPherson, and K. Schulten. 2006. Molecular dynamics simulations of the complete satellite tobacco mosaic virus. *Structure.* 14:437–449.

[5] Arkhipov, A., P. L. Freddolino, and K. Schulten. 2006. Stability and dynamics of virus capsids described by coarse-grained modeling. *Structure.* 14:1767–1777.

[6] Le, L., E. H. Lee, K. Schulten, and T. Truong. 2010. Molecular modeling of swine influenza A/H1N1, Spanish H1N1, and avian H5N1 flu N1 neuraminidases bound to Tamiflu and Relenza. *PLoS Currents: Influenza.* 2009 Aug 27:RRN1015. (9 pages).

[7] Le, L., E. H. Lee, D. J. Hardy, T. N. Truong, and K. Schulten. 2010. Molecular dynamics simulations suggest that electrostatic funnel directs binding of Tamiflu to influenza N1 neuraminidases. *PLoS Comput. Biol.* 6:e1000939. (13 pages).

[8] Lim, B., E. H. Lee, M. Sotomayor, and K. Schulten. 2008. Molecular basis of fibrin clot elasticity. *Structure.* 16:449–459.

[9] Tavoosi, N., R. L. Davis-Harrison, T. V. Pogorelov, Y. Z. Ohkubo, M. J. Arcario, M. C. Clay, C. M. Rienstra, E. Tajkhorshid, and J. H. Morrissey. 2011. Molecular determinants of phospholipid synergy in blood clotting. *J. Biol. Chem.* 286:23247–23253.

[10] Ohkubo, Y. Z. and E. Tajkhorshid. 2008. Distinct structural and adhesive roles of Ca^{2+} in membrane binding of blood coagulation factors. *Structure.* 16:72–81.

[11] Ohkubo, Y. Z., J. H. Morrissey, and E. Tajkhorshid. 2010. Dynamical view of membrane binding and complex formation of human factor VIIa and tissue factor. *J Thromb. Haem.* 8:1044–1053.

[12] Morrissey, J. H., V. Pureza, R. L. Davis-Harrison, S. G. Sligar, C. M. Rienstra, A. Z. Kijac, Y. Z. Ohkubo, and E. Tajkhorshid. 2009. Protein membrane interactions: blood clotting on nanoscale bilayer. *J Thromb. Haem.* 7:169–172.

[13] Morrissey, J. H., R. L. Davis-Harrison, N. Tavoosi, K. Ke, V. Pureza, J. M. Boettcher, M. C. Clay, C. M. Rienstra, Y. Z. Ohkubo, T. V. Pogorelov, and E. Tajkhorshid. 2010. Protein-phospholipid interactions in blood clotting. *Thromb. Res.* 125(Suppl. 1):S23–S25.

[14] Morrissey, J. H., V. Pureza, R. L. Davis-Harrison, S. G. Sligar, Y. Z. Ohkubo, and E. Tajkhorshid. 2008. Blood clotting reactions on nanoscale phospholipid bilayers. *Thrombosis Research.* 122:S23–S26.

[15] Interlandi, G. and W. Thomas. 2010. The catch bond mechanism between von Willebrand factor and platelet surface receptors investigated by molecular dynamics simulations. *Proteins: Struct., Func., Gen.* 78:2506–2522.

[16] Miller, Y., B. Ma, and R. Nussinov. 2011. The unique Alzheimers β-amyloid triangular fibril has a cavity along the fibril axis under physiological conditions. *J. Am. Chem. Soc.* 133:2742–2748.

[17] Parthasarathy, S., F. Long, Y. Miller, Y. Xiao, D. McElheny, K. Thurber, B. Ma, R. Nussinov, and Y. Ishii. 2011. Molecular-level examination of Cu^{2+} binding structure for amyloid fibrils of 40-residue Alzheimer's β by solid-state NMR spectroscopy. *J. Am. Chem. Soc.* 133:3390–3400.

[18] Miller, Y., B. Ma, C.-J. Tsai, and R. Nussinov. 2010. Hollow core of Alzheimers aβ42 amyloid observed by cryoEM is relevant at physiological pH. *Proc. Natl. Acad. Sci. USA.* 107:14128–14133.

[19] Fogolari, F., A. Corazza, P. Viglino, P. Zuccato, L. Pieri, P. Faccioli, V. Bellotti, and G. Esposito. 2007. Molecular dynamics simulation suggests possible interaction patterns at early steps of β2-microglobulin aggregation. *Biophys. J.* 92:1673 – 1681.

[20] Buchete, N.-V. and G. Hummer. 2007. Structure and dynamics of parallel β-sheets, hydrophobic core, and loops in Alzheimer's aβ fibrils. *Biophys. J.* 92:3032 – 3039.

[21] Buchete, N.-V., R. Tycko, and G. Hummer. 2005. Molecular dynamics simulations of Alzheimer's β-amyloid protofilaments. *J. Mol. Biol.* 353:804 – 821.

[22] Shaw, D. E., R. O. Dror, J. K. Salmon, J. P. Grossman, K. M. Mackenzie, J. A. Bank, C. Young, M. M. Deneroff, B. Batson, K. J. Bowers, E. Chow, M. P. Eastwood, D. J. Ierardi, J. L. Klepeis, J. S. Kuskin, R. H. Larson, K. Lindorff-Larsen, P. Maragakis, M. A. Moraes, S. Piana, Y. Shan, and B. Towles. 2009. Millisecond-scale molecular dynamics simulations on anton. In *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*, SC '09. ACM, New York, NY, USA, pages 39:1–39:11.

[23] DePamphilis, M. and J. Alder. 1971. Purification of Intact Flagella from *Escherichia coli* and *Bacillus subtilis*. *J. Bacteriol.* 105:376–383.

[24] Blair, D. F. 2003. Flagellar movement driven by proton translocation. *FEBS Lett.* 545:86–95.

[25] Yonekura, K., S. Maki-Yonekura, and K. Namba. 2005. Building the Atomic Model for the Bacterial Flagellar Filament by Electron Cryomicroscopy and Image Analysis. *Structure.* 13:407–412.

[26] Arkhipov, A., P. L. Freddolino, K. Imada, K. Namba, and K. Schulten. 2006. Coarse-grained molecular dynamics simulations of a rotating bacterial flagellum. *Biophys. J.* 91:4589–4597.

[27] Iino, T. 1974. Assembly of salmonella flagellin *in vitro* and *in vivo*. *J. Supramol. Struct.* 2:372–384.

[28] Yonekura, K., S. Maki-Yonekura, and K. Namba. 2003. Complete atomic model of the bacterial flagellar filament by electron cryomicroscopy. *Nature.* 424:643–650.

[29] Minamino, T., Y. Saijo-Hamano, Y. Furukawa, B. Gonzalez-Pedrajo, R. Macnab, and K. Namba. 2004. Domain organization and function of *salmonella* FliK, a flagellar hook-length control protein. *J. Mol. Biol.* 341:491–502.

[30] Hirano, T., T. Minamino, K. Namba, and R. Macnab. 2003. Substrate Specificity Classes and the Recognition Signal for *Salmonella* Type III Flagellar Export. *J. Bacteriol.* 185:2485–2492.

[31] Blocker, A., K. Komoriya, and S. Aizawa. 2003. Type III secretion systems and bacterial flagella: Insights into function from structural similarities. *Proc. Natl. Acad. Sci. USA.* 100:3027–3030.

[32] Namba, K. 2001. Roles of partly unfolded conformations in macromolecular self-assembly. *Genes to Cells.* 6:1–12.

[33] Chng, C. and A. Kitao. 2008. Thermal Unfolding Simulations of Bacterial Flagellin: Insight into its Refolding Before Assembly. *Biophys. J.* 94:3858–3871.

[34] Daura, X., A. E. Mark, and W. F. van Gunsteren. 1999. Peptide folding simulations: no solvent required? *Comput. Phys. Commun.* 123:97–102.

[35] Daidone, I., M. B. Ulmschneider, A. D. Nola, A. Amadei, and J. C. Smith. 2007. Dehydration-driven solvent exposure of hydrophobic surfaces as a driving force in peptide folding. *Proc. Natl. Acad. Sci. USA.* 104:15230–15235.

[36] Brooks, B. R., R. E. Bruccoleri, B. D. Olafson, D. J. States, S. Swaminathan, and M. Karplus. 1983. CHARMM: A program for macromolecular energy, minimization, and dynamics calculations. *J. Comp. Chem.* 4:187–217.

[37] Brooks, B. R., C. L. Brooks, A. D. Mackerell, L. Nilsson, R. J. Petrella, B. Roux, Y. Won, G. Archontis, C. Bartels, S. Boresch, A. Caflisch, L. Caves, Q. Cui, A. R. Dinner, M. Feig, S. Fischer, J. Gao, M. Hodoscek, W. Im, K. Kuczera, T. Lazaridis, J. Ma, V. Ovchinnikov, E. Paci, R. W. Pastor, C. B. Post, J. Z. Pu, M. Schaefer, B. Tidor, R. M. Venable, H. L. Woodcock, X. Wu, W. Yang, D. M. York, and M. Karplus. 2009. Charmm: The biomolecular simulation program. *J. Comp. Chem.* 30:1545–1614.

[38] Hess, B., C. Kutzner, D. van der Spoel, and E. Lindahl. 2008. Gromacs 4: Algorithms for highly efficient, load-balanced, and scalable molecular simulation. *J. Chem. Theor. Comp.* 4:435–447.

[39] Larsson, P. and E. Lindahl. 2010. A high-performance parallel-generalized born implementation enabled by tabulated interaction rescaling. *J. Comp. Chem.* 31:2593–2600.

[40] Pearlman, D. A., D. A. Case, J. W. Caldwell, W. S. Ross, T. E. Cheatham, S. DeBolt, D. Ferguson, G. Seibel, and P. Kollman. 1995. Amber, a package of computer programs for applying molecular mechanics, normal mode analysis, molecular dynamics and free energy calculations to simulate the structural and energetic properties of molecules. *Comput. Phys. Commun.* 91:1–41.

[41] Phillips, J. C., R. Braun, W. Wang, J. Gumbart, E. Tajkhorshid, E. Villa, C. Chipot, R. D. Skeel, L. Kale, and K. Schulten. 2005. Scalable molecular dynamics with NAMD. *J. Comp. Chem.* 26:1781–1802.

[42] Tanner, D. E., W. Ma, Z. Chen, and K. Schulten. 2011. Theoretical and computational investigation of flagellin translocation and bacterial flagellum growth. *Biophys. J.* 100:2548–2556.

[43] Tanner, D. E., K.-Y. Chan, J. Phillips, and K. Schulten. 2011. Parallel generalized Born implicit solvent calculations with NAMD. *J. Chem. Theor. Comp.* 7:3635–3642.

[44] Cornish, P. V., D. N. Ermolenko, H. F. Noller, and T. Ha. 2008. Spontaneous intersubunit rotation in single ribosomes. *Mol. Cell.* 30:578–588.

[45] Cornish, P., D. N. Ermolenko, D. W. Staple, L. Hoang, R. P. Hickerson, H. F. Noller, and T. Ha. 2009. Following movement of the L1 stalk between three functional states in single ribosomes. *Proc. Natl. Acad. Sci. USA.* 106:2571–2576.

[46] Horan, L. H. and H. F. Noller. 2007. Intersubunit movement is required for ribosomal translocation. *Proc. Natl. Acad. Sci. USA.* 104:4881–4885.

[47] Agirrezabala, X., J. Lei, J. L. Brunelle, R. F. Ortiz-Meoz, R. Green, and J. Frank. 2008. Visualization of the hybrid state of tRNA binding promoted by spontaneous ratcheting of the ribosome. *Mol. Cell.* 32:190–197.

[48] Stone, J. E., D. J. Hardy, B. Isralewitz, and K. Schulten. 2011. GPU algorithms for molecular modeling. In J. Dongarra, D. A. Bader, and J. Kurzak, editors, *Scientific Computing with Multicore and Accelerators*, chapter 16. Chapman & Hall/CRC Press, pages 351–371.

[49] Hardy, D. J., J. E. Stone, K. L. Vandivort, D. Gohara, C. Rodrigues, and K. Schulten. 2011. Fast molecular electrostatics algorithms on GPUs. In W. Hwu, editor, *GPU Computing Gems*, chapter 4. Morgan Kaufmann Publishers, pages 43–58.

[50] Stone, J. E., D. J. Hardy, J. Saam, K. L. Vandivort, and K. Schulten. 2011. GPU-accelerated computation and interactive display of molecular orbitals. In W. Hwu, editor, *GPU Computing Gems*, chapter 1. Morgan Kaufmann Publishers, pages 5–18.

[51] Roberts, E., J. E. Stone, L. Sepulveda, W. W. Hwu, and Z. Luthey-Schulten. 2009. Long time-scale simulations of in vivo diffusion using GPU hardware. In *Proceedings of the IEEE International Parallel & Distributed Processing Symposium*. pages 1–8.

[52] Still, W. C., A. Tempczyk, R. C. Hawley, and T. Hendrickson. 1990. Semianalytical treatment of solvation for molecular mechanics and dynamics. *J. Am. Chem. Soc.* 112:6127–6129.

[53] Onufriev, A., D. Bashford, and D. A. Case. 2004. Exploring protein native states and large-scale conformational changes with a modified generalized Born model. *Proteins: Struct., Func., Bioinf.* 55:383–394.

[54] Weiser, J., P. S. Shenkin, and W. C. Still. 1998. Approximate atomic surfaces from linear combinations of pairwise overlaps (lcpo). *J. Comp. Chem.* 20:217–230.

[55] Gumbart, J. and K. Schulten. 2008. The roles of pore ring and plug in the SecY protein-conducting channel. *J. Gen. Physiol.* 132:709–719.

[56] Neupert, W. and J. M. Herrmann. 2007. Translocation of Proteins into Mitochondria. *Annu. Rev. Biochem.* 76:723–749.

[57] Sung, W. and P. J. Park. 1996. Polymer Translocation through a Pore in a Membrane. *Phys. Rev. Lett.* 77:783–786.

[58] Schatz, G. and B. Dobberstein. 1996. Common Principles of Protein Translocation Across Membranes. *Science*. 271:1519–1526.

[59] Lubensky, D. K. and D. R. Nelson. 1999. Driven polymer translocation through a narrow pore. *Biophys. J.* 77:1824–1838.

[60] Dubbeldam, J., A. Milchev, V. Rostiashvili, and T. Vilgis. 2007. Driven polymer translocation through a nanopore: A manifestation of anomalous diffusion. *Europhys. Lett.* 79:18002.

[61] Simon, S., C. Peskin, and G. Oster. 1992. What drives the translocation of proteins? *Proc. Natl. Acad. Sci. USA*. 89:3770–3774.

[62] Wickner, W. and R. Schekman. 2005. Protein translocation across biological membranes. *Science*. 310:1452–1456.

[63] Muthukumar, M. and C. Y. Kong. 2006. Simulation of polymer translocation through protein channels. *Proc. Natl. Acad. Sci. USA*. 103:5273–5278.

[64] Loebl, H. C., R. Randel, S. P. Goodwin, and C. C. Matthai. 2003. Simulation studies of polymer translocation through a channel. *Phys. Rev. E*. 67:041913.

[65] Namba, K., I. Yamashita, and F. Vonderviszt. 1989. Structure of the core and central channel of bacterial flagella. *Nature*. 342:648–654.

[66] Mimori, Y., I. Yamashita, K. Murata, Y. Fujiyoshi, K. Yonekura, C. Toyoshima, and K. Namba. 1995. The structure of the R-type straight flagellar filament of *Salmonella* at 9 Å resolution by electron cryomicroscopy. *J. Mol. Biol.* 249:69–87.

[67] Samatey, F. A., K. Imada, S. Nagashima, F. Vonderviszt, T. Kumasaka, M. Yamamoto, and K. Namba. 2001. Structure of the bacterial flagellar protofilament and implications for a switch for supercoiling. *Nature.* 410:331–337.

[68] Ghosh, P. 2004. Process of Protein Transport by the Type III Secretion System. *Microbiol. Mol. Biol. Rev.* 68:771–795.

[69] Maki-Yonekura, S., K. Yonekura, and K. Namba. 2003. Domain movements of HAP2 in the cap-filament complex formation and growth process of the bacterial flagellum. *Proc. Natl. Acad. Sci. USA.* 100:15528–15533.

[70] Levy, E. M. 1973. Flagellar Elongation: An Example of Controlled Growth. *J. Theor. Biol.* 43:133–149.

[71] Levy, E. M. 1974. Flagellar Elongation as a Moving Boundary Problem. *Bull. Math. Biol.* 36:265–273.

[72] Mate, C. M., G. M. CcClelland, R. Erlandsson, and S. Chiang. 1987. Atomic-Scale Friction of a Tungsten Tip on a Graphite Surface . *Phys. Rev. Lett.* 59:1942–1945.

[73] Fusco, C. and A. Fasolino. 2005. Velocity dependence of atomic-scale friction: A comparative study of the one- and two-dimensional Tomlinson model. *Phys. Rev. B.* 71:045413.

[74] Heslot, F., T. Baumberger, B. Perrin, B. Caroli, and C. Caroli. 1994. Creep, stick-slip, and dry-friction dynamics: Experiments and a heuristic model. *Phys. Rev. E.* 49:4973–4988.

[75] Baumberger, T. and L. Gauthier. 1996. Creeplike Relaxation at the Interface between Rough Solids under Shear . *J. Phys. I France.* 6:1021–1030.

[76] Muser, M. H. 2008. How static is static friction? *Proc. Natl. Acad. Sci. USA.* 105:13187–13188.

[77] Yamakawa, H. 1971. *Modern Theory of Polymer Solutions.* Harper and Row, New York.

[78] Kawakatsu, T. 2004. *Statistical Physics of Polymers : An Introduction.* Springer, Berlin.

[79] Chemla, Y. R., K. Aathavan, J. Michaelis, S. Grimes, P. J. Jardine, D. L. Anderson, and C. Bustamante. 2005. Mechanism of Force Generation of a Viral DNA Packaging Motor. *Cell.* 122:683–692.

[80] Purohit, P. K., M. M. Inmdar, P. D. Grauson, T. M. Squires, J. Kondev, and R. Phillips. 2005. Forces during Bacteriophage DNA Packaging and Ejection. *Biophys. J.* 88:851–866.

[81] Humphrey, W., A. Dalke, and K. Schulten. 1996. VMD – Visual Molecular Dynamics. *J. Mol. Graphics.* 14:33–38.

[82] MacKerell, A. D., Jr., D. Bashford, M. Bellott, R. L. Dunbrack, Jr., J. D. Evanseck, M. J. Field, S. Fischer, J. Gao, H. Guo, S. Ha, D. Joseph, L. Kuchnir, K. Kuczera, F. T. K. Lau, C. Mattos, S. Michnick, T. Ngo, D. T. Nguyen, B. Prodhom, I. W. E. Reiher, B. Roux, M. Schlenkrich, J. Smith, R. Stote, J. Straub, M. Watanabe, J. Wiorkiewicz-Kuczera, D. Yin, and M. Karplus. 1998. All-atom empirical potential for molecular modeling and dynamics studies of proteins. *J. Phys. Chem. B.* 102:3586–3616.

[83] Jorgensen, W. L., J. Chandrasekhar, J. D. Madura, R. W. Impey, and M. L. Klein. 1983. Comparison of simple potential functions for simulating liquid water. *J. Chem. Phys.* 79:926–935.

[84] Morgan, D., C. Owen, L. Melanson, and D. DeRosier. 1995. Structure of Bacterial Flagellar Filaments at 11 Å Resolution: Packing of the α-Helices. *J. Mol. Biol.* 249:88–110.

[85] Isralewitz, B., M. Gao, and K. Schulten. 2001. Steered molecular dynamics and mechanical functions of proteins. *Curr. Opin. Struct. Biol.* 11:224–230.

[86] Kellermayer, M., S. Smith, H. Granzier, and C. Bustamante. 1997. Folding-unfolding transition in single titin modules characterized with laser tweezers. *Science.* 276:1112–1116.

[87] Li, H., W. Linke, A. F. Oberhauser, M. Carrion-Vazquez, J. G. Kerkvliet, H. Lu, P. E. Marszalek, and J. M. Fernandez. 2002. Reverse engineering of the giant muscle protein titin. *Nature.* 418:998–1002.

[88] Lee, G., K. Abdi, Y. Jiang, P. Michaely, V. Bennett, and P. E. Marszalek. 2006. Nanospring behaviour of ankyrin repeats. *Nature.* 440:246–249.

[89] Cantor, C. R. and P. R. Schimmel. 1980. *Biophysical Chemistry*, volume III. W. H. Freeman and Company, New York.

[90] Roberts, E., J. Eargle, D. Wright, and Z. Luthey-Schulten. 2006. MultiSeq: Unifying sequence and structure data for evolutionary analysis. *BMC Bioinformatics.* 7:382.

[91] Berendsen, H. J. C., J. P. M. Postma, W. F. van Gunsteren, and J. Hermans. 1981. Interaction models for water in relation to protein hydration. In B. Pullman, editor, *Intermolecular Forces.* D. Reidel Publishing Company, pages 331–342.

[92] Hassan, S. A. and E. L. Mehler. 2002. A critical analysis of continuum electrostatics: the screened Coulomb potential-implicit solvent model and the study of the alanine dipeptide and discrimination of misfolded structures of proteins. *Proteins: Struct., Func., Gen.* 47:45–61.

[93] Holst, M., N. Baker, and F. Wang. 2000. Adaptive multilevel finite element solution of the Poisson–Boltzmann equation II. refinement at solvent-accessible surfaces in biomolecular systems. *J. Comp. Chem.* 21:1343–1352.

[94] Nandi, N. and B. Bagchi. 1997. Dielectric relaxation of biological water. *J. Phys. Chem. B.* 101:10954–10961.

[95] Rhee, Y. M. and V. S. Pande. 2008. Solvent viscosity dependence of the protein folding dynamics. *J. Phys. Chem. B.* 112:6221–6227.

[96] Lu, B., X. Cheng, J. Huang, and J. A. McCammon. 2010. AFMPB: An adaptive fast multipole Poisson-Boltzmann solver for calculating electrostatics in biomolecular systems. *Comput. Phys. Commun.* 181:1150–1160.

[97] Baker, N. A. 2005. Improving implicit solvent simulations: a Poisson-centric view. *Curr. Opin. Struct. Biol.* 15:137–143.

[98] Hassan, S. A., E. L. Mehler, D. Zhang, and H. Weinstein. 2003. Molecular dynamics simulations of peptides and proteins with a continuum electrostatic model based on screened Coulomb potentials. *Proteins: Struct., Func., Gen.* 51:109–125.

[99] Schaefer, M. and M. Karplus. 1996. A comprehensive analytical treatment of continuum electrostatics. *J. Phys. Chem.* 100:1578–1599.

[100] Qiu, D., P. S. Shenkin, F. P. Hollinger, and W. C. Still. 1997. The gb/sa continuum model for solvation. a fast analytical method for the calculation of approximate born radii. *J. Phys. Chem.* 101:3005–3014.

[101] Feig, M., A. Onufriev, M. S. Lee, W. Im, D. A. Case, and C. L. Brooks. 2004. Performance comparison of generalized born and poisson methods in the calculation of electrostatic solvation energies for protein structures. *J. Comp. Chem.* 25:265–284.

[102] Lee, M. S., F. R. Salsbury, and C. L. Brooks. 2002. A high-performance parallel-generalized born implementation enabled by tabulated interaction rescaling. *J. Chem. Phys.* 116:10606–10614.

[103] Im, W., M. S. Lee, and C. L. Brooks. 2003. Generalized born model with a simple smoothing function. *J. Comp. Chem.* 24:1691–1702.

[104] Shivakumar, D., Y. Deng, and B. Roux. 2009. Computations of absolute solvation free energies of small molecules using explicit and implicit solvent model. *J. Chem. Theor. Comp.* 5:919–930.

[105] Grant, B. J., A. A. Gorfe, and J. A. McCammon. 2010. Large conformational changes in proteins: signaling and other functions. *Theoret. Chim. Acta.* 20:142–147.

[106] Schulz, R., B. Lindner, L. Petridis, and J. C. Smith. 2009. Scaling of multimillion-atom biological molecular dynamics simulation on a petascale supercomputer. *J. Chem. Theor. Comp.* 5:2798–2808.

[107] Bondi, A. 1964. van der waals volumes and radii. *J. Phys. Chem.* 68:441–451.

[108] Srinivasan, J., M. W. Trevathan, P. Beroza, and D. A. Case. 1999. Application of a pairwise generalized born model to proteins and nucleic acids: inclusion of salt effects. *Theoret. Chim. Acta.* 101:426–434.

[109] Onufriev, L., D. A. Case, and D. Bashford. 2002. Effective Born Radii in the Generalized Born Approximation: The Importance of Being Perfect. *J. Comp. Chem.* 23:1297–1304.

[110] Feig, M. and C. L. Brooks III. 2004. Recent advances in the development and application of implicit solvent models in biomolecule simulations. *Curr. Opin. Struct. Biol.* 14:217–224.

[111] Hawkins, G. D., C. J. Cramer, and D. G. Truhlar. 1996. Parameterized models of aqueous free energies of solvation based on pairwise descreening of solute atomic charges from a dielectric medium. *J. Phys. Chem.* 100:19824–19839.

[112] Schaefer, M. and C. Froemmel. 1990. A precise analytical method for calculating the electrostatic energy of macromolecules in aqueous solution. *J. Mol. Biol.* 216:1045 – 1066.

[113] Onufriev, A., D. Bashford, and D. A. Case. 2000. Modification of the generalized Born model suitable for macromolecules. *J. Phys. Chem.* 104:3712–3720.

[114] Kalé, L. V. and S. Krishnan. 1996. Charm++: Parallel programming with message-driven objects. In G. V. Wilson and P. Lu, editors, *Parallel Programming using C++*. MIT Press, pages 175–213.

[115] Kalé, L., R. Skeel, M. Bhandarkar, R. Brunner, A. Gursoy, N. Krawetz, J. Phillips, A. Shinozaki, K. Varadarajan, and K. Schulten. 1999. NAMD2: Greater scalability for parallel molecular dynamics. *J. Comp. Phys.* 151:283–312.

[116] Darden, T., D. York, and L. Pedersen. 1993. Particle mesh Ewald: An $N\cdot\log(N)$ method for Ewald sums in large systems. *J. Chem. Phys.* 98:10089–10092.

[117] Trabuco, L. G., E. Villa, K. Mitra, J. Frank, and K. Schulten. 2008. Flexible fitting of atomic structures into electron microscopy maps using molecular dynamics. *Structure.* 16:673–683.

[118] Wells, D. B., V. Abramkina, and A. Aksimentiev. 2007. Exploring transmembrane transport through α-hemolysin with grid-steered molecular dynamics. *J. Chem. Phys.* 127:125101.

[119] Villa, E., J. Sengupta, L. G. Trabuco, J. LeBarron, W. T. Baxter, T. R. Shaikh, R. A. Grassucci, P. Nissen, M. Ehrenberg, K. Schulten, and J. Frank. 2009. Ribosome-induced changes in elongation factor Tu conformation control GTP hydrolysis. *Proc. Natl. Acad. Sci. USA.* 106:1063–1068.

[120] Seidelt, B., C. A. Innis, D. N. Wilson, M. Gartmann, J.-P. Armache, E. Villa, L. G. Trabuco, T. Becker, T. Mielke, K. Schulten, T. A. Steitz, and R. Beckmann. 2009. Structural insight into nascent polypeptide chain-mediated translational stalling. *Science.* 326:1412–1415.

[121] Becker, T., S. Bhushan, A. Jarasch, J.-P. Armache, S. Funes, F. Jossinet, J. Gumbart, T. Mielke, O. Berninghausen, K. Schulten, E. Westhof, R. Gilmore, E. C. Mandon, and R. Beckmann. 2009. Structure of monomeric yeast and mammalian Sec61 complexes interacting with the translating ribosome. *Science.* 326:1369–1373.

[122] Agirrezabala, X., E. Schreiner, L. G. Trabuco, J. Lei, R. F. Ortiz-Meoz, K. Schulten, R. Green, and J. Frank. 2011. Structural insights into cognate vs. near-cognate discrimination during decoding. *EMBO J.* 30:1497–1507.

[123] Frauenfeld, J., J. Gumbart, E. O. van der Sluis, S. Funes, M. Gartmann, B. Beatrix, T. Mielke, O. Berninghausen, T. Becker, K. Schulten, and R. Beckmann. 2011. Cryo-EM structure of the ribosome-SecYE complex in the membrane environment. *Nat. Struct. Mol. Biol.* 18:614–621.

[124] Trabuco, L. G., E. Schreiner, J. Eargle, P. Cornish, T. Ha, Z. Luthey-Schulten, and K. Schulten. 2010. The role of L1 stalk-tRNA interaction in the ribosome elongation cycle. *J. Mol. Biol.* 402:741–760.

[125] Berk, V., W. Zhang, R. D. Pai, and J. H. D. Cate. 2006. Structural basis for mRNA and tRNA positioning on the ribosome. *Proc. Natl. Acad. Sci. USA*. 103:15830–15834.

[126] Cornell, W. D., P. Cieplak, C. I. Bayly, I. R. Gould, K. M. Merz, Jr., D. M. Ferguson, D. C. Spellmeyer, T. Fox, J. W. Caldwell, and P. A. Kollman. 1995. A second generation force field for the simulation of proteins, nucleic acids, and organic molecules. *J. Am. Chem. Soc.* 117:5179–5197.

[127] Hornak, V., R. Abel, A. Okur, B. Strockbine, A. Roitberg, and C. Simmerling. 2006. Comparison of multiple Amber force fields and development of improved protein backbone parameters. *Proteins*. 65:712–725.

[128] Perez, A., I. Marchn, D. Svozil, J. Sponer, T. E. Cheatham, C. A. Laughton, and M. Orozco. 2007. Refinement of the AMBER Force Field for Nucleic Acids: Improving the Description of α/γ Conformers. *Biophys. J.* 92:3817–3829.

[129] Aduri, R., B. T. Psciuk, P. Saro, H. Taniga, H. B. Schlegel, and J. SantaLucia. 2007. AMBER force field parameters for the naturally occurring modified nucleosides in RNA. *J. Chem. Theor. Comp.* 3:1464–1475.

[130] Stella, L. and S. Melchionna. 1998. Equilibration and sampling in molecular dynamics simulations of biomolecules. *J. Chem. Phys.* 109:10115–10117.

[131] Wower, I. K., J. Wower, and R. A. Zimmermann. 1998. Ribosomal protein L27 participates in both 50S subunit assembly and the peptidyl transferase reaction. *J. Biol. Chem.* 273:19847–19852.

[132] Cukras, A. R., D. R. Southworth, J. L. Brunelle, G. M. Culver, and R. Green. 2003. Ribosomal proteins S12 and S13 function as control elements for translocation of the mRNA:tRNA complex. *Mol. Cell.* 12:321–328.

[133] Diaconu, M., U. Kothe, F. Schlünzen, N. Fischer, J. M. Harms, A. G. Tonevitsky, H. Stark, M. V. Rodnina, and M. C. Wahl. 2005. Structural basis for the function of the ribosomal L7/12 stalk in factor binding and GTPase activation. *Cell.* 121:991–1004.

[134] Cheng, L. S., R. E. Amaro, D. Xu, W. W. Li, P. W. Arzberger, and J. A. McCammon. 2008. Ensemble-based virtual screening reveals potential novel antiviral compounds for avian influenza neuraminidase. *J. Med. Chem.* 51:3878–3894.

[135] Acharya, R., V. Carnevale, G. Fiorin, B. G. Leviné, A. L. Polishchuk, V. Balannik, I. Samish, R. A. Lamb, L. H. Pinto, W. F. DeGrado, and M. L. Klein. 2010. Structure and mechanism of proton transport through the transmembrane tetrameric M2 protein bundle of the influenza a virus. *Proc. Natl. Acad. Sci. USA*. 107:15075–15080.

[136] Khurana, E., M. D. Peraro, R. DeVane, S. Vemparala, W. F. DeGrado, and M. L. Klein. 2009. Molecular dynamics calculations suggest a conduction mechanism for the M2 proton channel from influenza A virus. *Proc. Natl. Acad. Sci. USA*. 106:1069–1074.

[137] Khurana, E., R. H. DeVane, M. D. Peraro, and M. L. Klein. 2011. Computational study of drug binding to the membrane-bound tetrameric M2 peptide bundle from influenza A virus. *Biochim. Biophys. Acta*. 1808:530–537.

[138] Newhouse, E. I., D. Xu, P. R. L. Markwick, R. E. Amaro, H. C. Pao, K. J. Wu, M. Alam, J. A. McCammon, and W. W. Li. 2009. Mechanism of glycan receptor recognition and specificity switch for avian, swine, and human adapted influenza virus hemagglutinins: A molecular dynamics perspective. *J. Am. Chem. Soc.* 131:17430–17442.

[139] Fidelak, J., J. Juraszek, D. Branduardi, M. Bianciotto, and F. L. Gervasio. 2010. Free-energy-based methods for binding profile determination in a congeneric series of CDK2 inhibitors. *J. Phys. Chem. B.* 114:9516–9524.

[140] Davidovich, C., A. Bashan, and A. Yonath. 2008. Structural basis for cross-resistance to ribosomal PTC antibiotics. *Proc. Natl. Acad. Sci. USA*. 105:20665–20670.

[141] Poehlsgaard, J. and S. Douthwaite. 2005. The bacterial ribosome as a target for antibiotics. *Nat. Rev. Microbiol.* 3:870–881.

[142] Hsin, J., J. Strümpfer, E. H. Lee, and K. Schulten. 2011. Molecular origin of the hierarchical elasticity of titin: simulation, experiment and theory. *Annu. Rev. Biophys.* 40:187–203.

[143] Venkatesan, B., J. Polans, J. Comer, S. Sridhar, D. Wendell, A. Aksimentiev, and R. Bashir. 2011. Lipid bilayer coated Al_2O_3 nanopore sensors: Towards a hybrid biological solid-state nanopore. *Biomed. Microdev.*:1–12. ISSN 1387-2176.

[144] Carr, R., J. Comer, M. D. Ginsberg, and A. Aksimentiev. 2011. Microscopic perspective on the adsorption isotherm of a heterogeneous surface. *The Journal of Physical Chemistry Letters*. 2:1804–1807. doi:10.1021/jz200749d.

[145] Mei, C., Y. Sun, G. Zheng, E. J. Bohm, L. V. Kalé, J. C. Phillips, and C. Harrison. 2011. Enabling and scaling biomolecular simulations of 100 million atoms on petascale machines with a multicore-optimized message-driven runtime. In *Proceedings of the 2011 ACM/IEEE conference on Supercomputing*. Seattle, WA.

[146] Stone, J. E., J. C. Phillips, P. L. Freddolino, D. J. Hardy, L. G. Trabuco, and K. Schulten. 2007. Accelerating molecular modeling applications with graphics processors. *J. Comp. Chem.* 28:2618–2640.

[147] Phillips, J. C., J. E. Stone, and K. Schulten. 2008. Adapting a message-driven parallel application to GPU-accelerated clusters. In *SC '08: Proceedings of the 2008 ACM/IEEE Conference on Supercomputing*. IEEE Press, Piscataway, NJ, USA. ISBN 978-1-4244-2835-9.

[148] Harvey, M. J., G. Giupponi, and G. D. Fabritiis. 2009. ACEMD: Accelerating biomolecular dynamics in the microsecond time scale. *J. Chem. Theor. Comp.* 5:1632–1639. ISSN 1549–9618. doi: http://dx.doi.org/10.1021/ct9000685.

[149] Baker, C. M., V. M. Anisimov, and A. D. MacKerell. 2011. Development of charmm polarizable force field for nucleic acid bases based on the classical drude oscillator model. *J. Phys. Chem. B.* 115:580–596. doi:10.1021/jp1092338.

[150] Friedrichs, M. S., P. Eastman, V. Vaidyanathan, M. Houston, S. Legrand, A. L. Beberg, D. L. Ensign, C. M. Bruns, and V. S. Pande. 2009. Accelerating molecular dynamic simulation on graphics processing units. *J. Comp. Chem.* 30:864–872.

[151] Eastman, P. and V. S. Pande. 2010. Efficient nonbonded interactions for molecular dynamics on a graphics processing unit. *J. Comp. Chem.* 31:1268–1272.

[152] Anderson, J. A., C. D. Lorenz, and A. Travesset. 2008. General purpose molecular dynamics simulations fully implemented on graphics processing units. *J. Chem. Phys.* 227:5342–5359. doi: http://dx.doi.org/10.1016/j.jcp.2008.01.047.

[153] Liu, F. and M. Gruebele. 2007. Tuning lambda6-85 towards downhill folding at its melting temperature. *J. Mol. Biol.* 370:574–584.

[154] Izrailev, S., S. Stepaniants, B. Isralewitz, D. Kosztin, H. Lu, F. Molnar, W. Wriggers, and K. Schulten. 1998. Steered molecular dynamics. In P. Deuflhard, J. Hermans, B. Leimkuhler, A. E. Mark, S. Reich, and R. D. Skeel, editors, *Computational Molecular Dynamics: Challenges, Methods, Ideas*, volume 4 of *Lecture Notes in Computational Science and Engineering*. Springer-Verlag, Berlin, pages 39–65.

[155] Case, D., T. Cheatham III, T. Darden, H. Gohlke, R. Luo, K. Merz Jr, A. Onufriev, C. Simmerling, B. Wang, and R. Woods. 2005. The amber biomolecular simulation programs. *J. Comp. Chem.* 26:1668.

[156] Sitkoff, D., K. A. Sharp, and B. Honig. 1994. Accurate calculation of hydration free energies using macroscopic solvent models. *J. Phys. Chem.* 98:1978–1988.

[157] Ryckaert, J.-P., G. Ciccotti, and H. J. C. Berendsen. 1977. Numerical integration of the Cartesian equations of motion of a system with constraints: Molecular dynamics of *n*-alkanes. *J. Comp. Phys.* 23:327–341.

[158] Zhu, J., Y. Shi, and H. Liu. 2002. Parametrization of a generalized Born/solvent-accessible surface area model and applications to the simulation of protein dynamics. *J. Phys. Chem. B.* 106:4844–4853.

[159] Kalé, L. V., G. Zheng, C. W. Lee, and S. Kumar. 2006. Scaling applications to massively parallel machines using projections performance analysis tool. *Fut. Gen. Comp. Sys.* 22:347–358.

[160] Carslaw, H. and J. Jaeger. 1959. *Conduction of Heat in Solids*. Oxford University Press, Oxford, England, 2nd edition.